Università degli Studi di Padova

Facoltà di Ingegneria

Corso di Laurea Specialistica in Ingegneria Informatica

tesi di laurea

Progettazione e sviluppo di un Web Feature Service per dati geospaziali in formato CityGML basato su strumenti opensource.

Relatore: prof. Massimo Rumor

Laureando: Daniele Toniolo

18 aprile 2011

anno accademico 2010/2011

Alla mia famiglia
in particolare ai miei genitori
per avermi sempre sostenuto.

Sommario

L'OGC (Open Geospatial Consortium) ha prodotto due specifiche per la strutturazione dei dati urbani in tre dimensioni, denominate CityGML e KML ed ha anche standardizzato, in una architettura SOA, un insieme di servizi dedicati ai dati spaziali in due dimensioni.
Sono in discussione, ma non sono state ancora emanate specifiche per gli stessi servizi per i dati tridimensionali. Il presente lavoro riguarda proprio lo sviluppo di un servizio web, per l'accesso a feature tridimensionali progettato in analogia all'omologo servizio WFS standardizzato per i dati bidimensionali, con la possibilità di personalizzare la vestizione grafica degli oggetti 3D per mezzo dello standard SLD. Il servizio è stato sviluppato come estensione del server geospaziale FOSS Geoserver.

Indice

Capitolo 1

Premessa

L'utilizzo di modelli tridimensionali è in aumento, come si rileva dallo sviluppo dei ben noti applicativi Google Earth e Virtual Earth.

L'OGC (Open Geospatial Consortium) ha prodotto due specifiche per la strutturazione dei dati urbani in tre dimensioni, denominate CityGML e KML e questo ha stimolato la realizzazione di strumenti ed applicazioni desktop per visualizzare ed analizzare dati 3D.

I due modelli standard sono definiti entrambi in formato XML. KML è un modello puramente geometrico, CityGML invece comprende oltre alla geometria anche la topologia, le appearance e l'informazione semantica associata agli oggetti 3D.

Sempre OGC, a fronte dello sviluppo dei sistemi GIS e della necessità di aumentare il livello di interoperabilità, ha standardizzato, in una architettura SOA, un insieme di servizi dedicati ai dati spaziali in due dimensioni.

In questo contesto le infrastrutture di dati territoriali forniscono una struttura appropriata per un accesso flessibile alle varie sorgenti di dati distribuite. L'utilizzo dei servizi standard OGC comporta due importanti vantaggi per gli utenti, del dato geografico: l'inutilità di duplicazione dei dati in locale e la certezza di consultare/utilizzare una versione certificata dei dati richiesti in quanto il dato viene distribuito direttamente da chi lo produce o ne certifica l'attendibilità.

Sono in discussione, ma non sono state ancora emanate specifiche per gli stessi servizi per i dati tridimensionali.

Il presente lavoro riguarda lo sviluppo di un servizio web dedicato ai dati tridimensionali urbani.

Capitolo 2

Obiettivi

Nell'ambito di questa tesi è descritto il lavoro di progettazione e sviluppo di un Web Service per l'acceso ai dati vettoriali tridimensionali gestiti in un geoDBMS in analogia a WFS, standard OGC per i dati bidimensionali.
L'Interfaccia Standard OpenGIS Web Feature Service (WFS) permette la richiesta e l'importazione da parte di un client di oggetti geografici attraverso il Web, usando chiamate indipendenti dalla piattaforma. La codifica standard è il GML, basata su XML, ma anche altri formati possono essere usati per il trasporto delle informazioni.
Numerosi server geospaziali, come ad esempio Geoserver e Mapserver, implementano la componente WFS permettendo una comoda interfaccia per lo scambio di dati. Attualmente tuttavia l'implementazione è relativa alla sola parte bidimensionale.

Il servizio WFS 3D realizzato si occupa di rispondere alle richieste effettuate secondo lo standard WFS e di:

- fornire le geometrie richieste nel formato CityGML;
- tematizzare dinamicamente sulla base di parametri applicati in input;

Questo progetto utilizza le strutture dati e gli strumenti predisposti nell'ambito della tesi *"Progettazione e sviluppo di un toolkit per la gestione di dati spaziali 3D nei formati standard OGC CityGML e KML per il geodatabase opensource PostGIS."* [1].

Capitolo 3

Standard OGC

L'Open Geospatial Consortium è un consorzio internazionale no-profit costituito da aziende private, agenzie governative ed università che si prefigge di sviluppare delle regole standard per i servizi geospaziali.
Questi sono classificabili come sistemi software che scambiano dati sul protocollo HTTP. L'interazione tra i servizi e le applicazioni avviene mediante l'invio di messaggi XML. Il sistema di comunicazione garantisce l'interoperabilità essendo indipendente dalla piattaforma hardware, dal sistema operativo e dal formato originario dei dati: qualsiasi software client e server possono comunicare tra di loro purché implementino in modo corretto gli standard. L'interscambio di dati geo-spaziali avviene tra diversi elaboratori facenti parte della stessa rete o facenti parte di reti diverse comunicanti tra loro (World Wide Web). Tra gli standard più importanti si citano:

- Web Feature Service 3.1;
- Filter Encoding 3.2;
- Styled Layer Descriptor 3.3;
- CityGML 3.4.

3.1 OpenGIS® Web Feature Service

3.1.1 Funzioni ed operazioni principali

L'Interfaccia Standard OpenGIS Web Feature Service (WFS) permette la richiesta e l'importazione da parte di un client di oggetti geografici attraverso il Web, usando chiamate indipendenti dalla piattaforma.
Il protocollo WFS supporta le operazioni di INSERT, UPDATE, LOCK, QUERY

e DISCOVERY su feature geografiche usando HTTP come piattaforma di elaborazione distribuita. In termini generali il protocollo deve essere eseguito in ordine ai processi di richiesta WFS. Le elaborazioni dovrebbero procedere come segue:

1. Un'applicazione client dovrebbe richiedere un documento *capabilites* dal WFS. Tale documento contiene una descrizione di tutte le operazioni che il WFS supporta ad una lista di feature disponibili;

2. (Opzionale) Un'applicazione client effettua una richiesta al web feature service per la definizione di una o più tipi di elementi e feature;

3. Basandosi sulla definizione dei tipi di feature, l'applicazione client genera una richiesta;

4. La richiesta viene spedita al web server;

5. Il WFS è invocato per leggere e servire la richiesta inviata;

6. Quando il WFS ha completato di processare la richiesta, esso genererà una risposta la quale verrà restituita al client. Se nell'evento vi è accorso un errore, la risposta dovrà indicare tale errore.

Per supportare i processi di transazioni e interrogazioni, sono definite le seguenti operazioni:
GetCapabilities

Un web feature service può essere in grado di descrivere le proprie *capabilities*. Più in dettaglio dovrebbe indicare quali tipi di feature può servire e le operazioni supportate per ogni feature.

DescribeFeatureType

Un web feature service può essere in grado, su richiesta, di descrivere la struttura di ogni tipo di feature che può servire.

GetFeature

Un web feature service può essere in grado di rispondere ad una richiesta, reperendo l'istanza della feature. Inoltre, il client dovrebbe essere in grado di specificare quale proprietà delle feature estrarre, e di vincolare tramite query.

GetGMLObject

Un web feature service può essere in grado di servire una richiesta, e reperire le istanze degli elementi tramite l'attraversamento di XLink che riferiscono al loro XML ID.

Transaction

Un web feature service può essere in grado di servire una richiesta di transazione. Una richiesta di transazione composta di operazioni che modificano le feature, come operazioni per creare, aggiornare e eliminare feature geografiche.

LockFeature

Un web feature service può essere in grado di servire una richiesta di lock su una o più istanze di una feature type per la durata di una transizione. Questo assicura che la serializzazione delle transizioni è supportata.

Basandosi sulle operazioni descritte in precedenza tre classi di web feature service possono essere descritte:
Basic WFS

Un Basic WFS deve implementare le operazioni GetCapabilites, DescribeFeatureType e GetFeature. Questo si può considerare un web feature service di sola lettura.

XLink WFS

Un XLink WFS deve supportare tutte le operazioni di un basic web feature services ed in più deve implementare l'operazione GetGMLObject per XLink locali o remoti. Inoltre offre le opzioni per l'operazione GetGMLObject per essere performante durante le operazioni GetFeature.

Transaction WFS

Un Transaction WFS deve supportare tutte le operazioni di un Basic web feature services inoltre deve implementare le operazioni Transaction. Opzionalmente un Trasaction WFS può implementare le operazioni GetGMLObject e LockFeature.

3.1.2 Namespaces

I namespaces sono utilizzati per discriminare i dizionari XML gli uni dagli altri. Per il WFS ci sono tre definizioni di namespace che seguono la normativa, ossia:

- (http://www.opengeospatial.net/wfs) - per il dizionario dell'interfaccia WFS

- (http://www.opengeospatial.net/gml) - Per il dizionario GML
- (http://www.opengeospatial.net/ogc) - Per il dizionario dei filtri OGC

Una implementazione WFS potrebbe far uso di uno o più schemi di GML e questi schemi utilizzeranno, a loro volta, uno o più namespaces.

3.1.3 Operazione DescribeFeatureType

La funzione dell'operazione *DescribeFeatureType* è di generare uno schema di descrizione di una feature type richiesta ad un WFS. Lo schema di descrizione definisce le codifiche in ingresso delle feature (attraverso Insert e richieste di Update) e come le istanze delle feature verranno generate in uscita (in risposta a GetFeature ed a richieste GetGmlObject). L'unica risposta obbligatoria ad una richiesta *DescribeFeatureType* è uno *GML3 application schema* definito utilizzando XML Schema.

Richiesta

Un elemento *DescribeFeatureType* contiene zero o più elementi *TypeName* che codificano il nome di una feature type che devono essere descritte. Se il contenuto dell'elemento *DescribeFeatureType* è vuoto, allora questo verrà interpretato come richiesta di una descrizione di tutti i tipi di elementi che un WFS può servire. Il frammento di schema XML che definisce la codifica di una richiesta *DescribeFeatureType* è inA.1.1.

Risposta

In risposta ad una richiesta *DescribeFeatureType*, in cui il valore dell'attributo *outputFormat* è stato impostato a text/xml;subtype=gml/3.1.1, il WFS deve essere in grado di presentare un documento XML Schema valido e definendo lo schema della feature type elencate nella richiesta.

3.1.4 Operazione GetFeature

In GML una feature è rappresentata come un elemento XML. Il nome dell'elemento di una feature indica il *FeatureType* convenzionalmente dato in UpperCamelCase(Maiuscola), come xmml:BoreHole.
Il contenuto di una feature è un insieme di elementi, che descrivono la feature come un insieme di proprietà. Il nome di una proprietà indica il tipo di proprietà, convenzionalmente dato nel LowerCamelCase(minuscola), come

gml:boundedBy o xml:collarLocation.
Il valore di una proprietà è dato in linea con il contenuto dell'elemento, o per riferimento come valore di una risorsa identificata in un link effettuato come un attributo XML dell'elemento proprietà. Se la forma in linea è usata, allora il contenuto può essere un letterale (numero, testo, ...), o può essere costruito usando elementi XML, ma nessuna ipotesi può essere fatta circa la struttura del valore di una proprietà.
L'operazione GetFeature consente il recupero di feature da un web feature service. Una richiesta GetFeature viene elaborata da un WFS e quando il valore dell'attributo outputFormat è impostato a text/GML;subtype=gml/3.1.1, viene restituita al client una istanza del documento GML, contenente l'insieme dei risultati.

Richiesta

La codifica XML di una richiesta GetFeature viene esposta dal frammento di schema XML in A.1.2.
L'elemento <GetFeature> contiene uno o più elementi <query>, ognuno dei quali contiene la descrizione di una interrogazione. I risultati di tutte le query contenute in una richiesta GetFeature sono concatenate per produrre il set di risultati.
Gli attributi opzionali *outputFormat* specificano il formato della risposta alla richiesta GetFeature. Il valore di deafult è text/xml;subtype=gml/3.1.1 indicando che è un documento valido GML3. Per la retrocompatibilità, il valore GML2 o text/xml;subtype=gml/2.1.2 può essere specificato indicando che un documento in formato GML2 può essere generato.
Altri formati di output sono possibili per valori appropriati per l'attributo outputFormat che sono indicati nel documento GetCapabilities.

Un web feature service può rispondere ad una richiesta GetFeature in due modi. Può o generare un documento di risposta completo o più semplicemente ritornare un conteggio del numero di feature che una richiesta GetFeature dovrebbe ritornare. L'attributo opzionale resultType è usato per controllare come un web feature service risponde ad una richiesta GetFeature. I possibili valori per l'attributi elencati nella Tabella 3.1

L'attributo *maxFeature* limita il numero massimo di feature che possono essere restituite dopo una richiesta getFeature. Di default questo parametro non è settato e quindi vengono restituite tutte le feature senza limitazioni.
Ogni singola query contenuta in una richiesta GetFeature è definita usando l'elemento <query>. L'elemento <query> definisce quale *FeatureType* interrogare, quali proprietà reperire e quali vincoli applicare al fine di selezionare il

Valore	Descrizione
Results	Il valore di default, indica che il WFS deve generare come risposta l'intero documento
Hits	Il valore indica che il WFS ritorna la sola enumerazione delle feature che dovrebbe restituire

Tabella 3.1: Valori per l'attributo resultType

set delle feature valido.
L'attributo obbligatorio *typeName* è usato per indicare il nome di una o più istanze di FeatureType o istanze di classi che sono interrogate. Il suo valore è una lista di namespace-qualified, il cui valore deve corrispondere a uno dei FeatureType mostrato nel documento *Capabilities* del WFS. Indicare più di un typeName indica l'uso di una operazione di join. Tutti i nomi nella lista typeName devo essere tipi validi che appartengono al contenuto di questa feature query come definito dall'Application Schema GML.
L'attributo opzionale *featureVersion* su l'elemento <query> è utilizzato per favorire i sistemi che supportano il versioning. L'attributo opzionale *srsName* di un elemento <query> è usato per specificare uno specifico SRS(Spatial Reference System) supportato dal WFS da utilizzare per ritornare le feature geometriche.
Il valore di ogni elemento <wfs:PropertyName> è un namespace-qualified (ad esempio *myns:address*) il cui valore deve corrispondere ad una delle proprietà della feature pertinente. La feature pertinente è del tipo indicato nell'attributo typeName dell'elemento <Query>. Il nome degli elementi property possono essere scoperti dallo schema della feature type, ottenuto come risultato di una DescribeFeatureType.
La risposta ad un richiesta Getfeature deve essere in accordo alla struttura descritta dallo XML schema della FeatureType. Il WFS deve riportare tutte le proprietà obbligatorie di ogni feature, così come ogni proprietà richiesta attraverso l'elemento <wfs:PropertyName>. Nel caso che un WFS incontri una query che non seleziona tutte le proprietà obbligatorie di una feature, il WFS automaticamente dovrà aumentare la lista delle proprietà per includere tutti i nomi obbligatori. Un client deve essere in grado di trattare una situazione dove riceve più valori di quelli che ha richiesto attraverso l'elemento <wfs:PropertyName>.

L'elemento *Filter* è usato per definire i filtri sulla query. Possono essere specificati filtri spaziali e/o non spaziali e vengono descritti in 3.2].

Risposta

Il formato di una risposta ad una richiesta GetFeture è controllata attraverso l'attributo outputFormat. Il valore di default dell'attributo outputFormat è *text/xml; subtype=gml/3.1.1*. Questo indica che un WFS deve generare un documento GML che risulterà conforme alle OpenGis Geography Markup Language Implementation Specification [2], versione 3.1.1, e più specificatamente, l'output deve essere valido con lo schema generato dalla operazione DescribeFeatureType.

L'elemento radice di una risposta ad una operazione GetFeture deve essere l'elemento <wfs:FeatureCollection> il quale definito attraverso dal frammento di XML schema in A.1.3
Il contenuto dell'elemento <wfs:FeatureCollection> è controllato tramite il valore dell'attributo *resulType* sull'elemento <GetFeture>.

3.1.5 Operazione GetCapabilities

Ogni OGC Web Service (OWS), incluso un Web Feature Service, deve avere la capacità di descrivere le capabilities(capacità), attraverso un servizio di ritorno di metadati in risposta ad una richiesta GetCapabilities. Specificamente, ogni web feature service deve supportare la forma KVP che codifica la richiesta GetCapabilities su HTTP GET in modo che un client possa sempre conoscere come ottenere un documento capabilities.

Richiesta

L'elemento <GetCapabilities> è usato per richiedere un documento capabilities ad un web feature service. Di seguito viene definito lo schema XML per una richiesta:

```
<xsd:element name="GetCapabilities" type="wfs:
    GetCapabilitiesType"/>
<xsd:complexType name="GetCapabilitiesType">
  <xsd:complexContent>
    <xsd:extension base="ows:GetCapabilitiesType">
      <xsd:attribute name="service" type="ows:
          ServiceType" use="optional" default="WFS"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Risposta

L'elemento radice della risposta ad una richiesta <GetCapabilities> è l'elemento <WFS_Capabilities> ed è definito dal frammento di schema XML in A.1.4.

Dallo schema si può notare come il documento capabilities è suddiviso in sezioni.

Service Indentification La sezione ServiceIdentification, contiene metadati di base dello specifico server. La sezione ServiceIdentification comprende i paramatri e le parti descritte e specificate nella tabella Tabella 3.2

Sezione ServiceProvider La sezione ServiceProvider, contiene metadati circa l'organizzazione operarativa di questo server. La sezione ServiceProvider comprende i paramatri e le parti descritte e specificate nella tabella Tabella 3.3

Sezione OperationsMetadata La sezione OperationsMetadata contiene i metadati relativi alle operazioni fornite ed implementate dal servizio, inclusi gli URL per le varie operazioni supportate. La sezione OperationsMetadata comprende le sottosezioni descritte e specificate nella tabella Tabella 3.4.

Sezione FeatureTypeList La sezione FeatureTypeList definisce le operazioni comuni tra tutti i tipi di feature e la descrizione di ogni feature che il servizio offre. La sezione FeatureTypeList comprende le sottosezioni descritte e specificate nella tabella Tabella 3.5 e Tabella 3.6 .

Sezione ServesGMLObjectType La sezione ServesGMLObjectType contiene una lista di nomi di oggetti di tipo GML che un web feature service, supportando l'operazione GetGMLObject, offre.

Sezione SupportGMLObjectTypeList Questa sezione definisce la lista di oggetti GML che un WFS server sarebbe in grado di elaborare;

Nomi	Definizione	Tipi di Dati	Molteplicità
serviceType	usato per le comunicazioni macchina-macchina	URN[b]	Uno(Obbligatorio)
version	Versione del service type implementato	String	Uno o più (obbligatorio) Uno per ogni versione implementata dal server (non ordinata)
profile	Identificatore di una application profile OWS	String	zero o più (opzionale)
title	titolo del server	LanguageString	uno o più (obbligatorio)
abstract	Breve descrizione di questo server, normalmente disponibile alla visualizzazione	String	Zero o pi (opzionale)
keywords	lista non ordinata di una o pi parola(e) o frase(i) comunemente usate o formalizzate per descrivere il server	MD_Keywords class in ISO 19115	zero o pi (Opzionale)
fees	tariffe e condizioni per l'utilizzo di questo server	stringa di caratteri non vuota	Zero o pi (Opzionale)
access constraint	vincoli e restrizioni sul recupero e utilizzo dei dati	String	zero o più (opzionale)

Tabella 3.2: Parametri inclusi in una sezione ServiceIdentification

Nomi	Definizione	Tipi di Dati	Molteplicità
providerName	identificatore dell'organizzazione	String	Uno (Obbligatorio)
providerSite	riferimento al sito web del fornitore del servizio	CI_OnlineResource class in ISO 19115	Zero o una (opzionale)
serviceContact	Informazioni per contattare il fornitore del servizio	CI_ResponsibleParty classes in ISO 19115	Zero o una (opzionale)

Tabella 3.3: Parametri inclusi in una sezione ServiceProvider

Nomi	Definizione	Molteplicità ed uso
operation	Metadati per le operazioni implementate	Uno o pi (Obbligatorio)
parameter	Parametrio per il dominio applicato all'operazione	Zero o una (opzionale)
constraint	Vincoli sui domin	Zero o una (opzionale)
extendedCapabilities	Metadati relativi al server e a software supplementari	Zero o una (opzionale)

Tabella 3.4: Parametri inclusi in una sezione OperationsMetadata

Nome Elemento	Descrizione
Insert	L'elemento <Insert> usato per indicare che il wfs è capace di creare nuove istanze di un future type.
Update	L'elemento <Update> indica che il WFS può cambiare lo stato esistente di una feature.
Delete	L'elemento <Delete> indica che il WFS può cancellare o rimuovere istanze di una future typt dal datastore.
Query	L'elemento <Query> indica che il WFS è capace di eseguire un interrogazione su un feature type
Lock	L'elemento <Lock> indica che il WFS è capace di fare il lock ad una istanza di una feature type.

Tabella 3.5: Azioni di transizione e interrogazione su Feature

Elemento	Descrizione
Name	Il namespace qualifica il nome della feature type. (obbligatorio)
Title	<Title> è un titolo leggibile da una persona per identificare brevemente la feature type.
Abstract	<Abstract> è una descrizione narrativa per più informazioni riguardanti la feature type.
Keyword	<Keyword> contiene parole per aiutare la ricerca.
DefaultSRS	<DefaultSRS> indica quale sistema di riferimento spaziale viene usato dal WFS in modo di default.
OtherSRS	<OtherSRS> usato per indicare altri SRS supportati.
NoSRS	<NoSRS> usato per feature type che non hanno proprietà spaziali.
Operations	<Operations> definisce quali operazioni sono supportate su una feature type. Ogni operazione definita localmente ha la precedenza sulle operazioni definite globalmente.
OutputFormats	Questa è una lista di MIME type indicando il formato di output che può essere generato.
WGS84BoundingBox	L'elemento <WGS84BoundingBox> è usato per indicare il Bounding Box in WGS84.
MetaDataURL	Un WFS può usare zero o più elementi <MetaDataURL> per offrire altri dettagli.

Tabella 3.6: Elementi per descrivere le FeatureType

3.2 OpenGIS® Filter Encoding

Una espressione di filtro è un costrutto usato per vincolare i valori delle proprietà di un tipo di oggetto al fine di identificare un sottoinsieme di istanze per essere. L'intento di questo capitolo è quello di descrivere la codifica XML del catalogo OGC Common Query Language (CQL) come una rappresentazione neutrale di una query. Utilizzando gli strumenti per XML, è possibile convalidare, analizzare e trasformare facilmente in un altro linguaggio per recuperare o modificare le istanze di oggetti conservati in un archivio e/o in un database. Per esempio, un documento XML codificato come filtro potrebbe essere trasformato in una clausola WHERE di un'istruzione SQL SELECT per recuperare i dati memorizzati in un database relazionale basato su SQL, allo stesso modo potrebbe essere trasformato in un XPath o espressione XPointer per il recupero dei dati da documenti XML. Una vasta classe di servizi web OpenGIS®, richiedono la capacità di esprimere le espressioni di filtro in XML.

3.2.1 Elementi Filter Encoding

L'elemento <filter> è l'elemento radice e contiene l'espressione che viene creata combinando gli elementi definiti dal seguente frammento XML Schema:

```
<xsd:complexType name="FilterType">
  <xsd:choice>
  <xsd:element ref="ogc:spatialOps"/>
  <xsd:element ref="ogc:comparisonOps"/>
  <xsd:element ref="ogc:logicOps"/>
  <xsd:element ref="ogc:_Id" maxOccurs="unbounded"/>
  </xsd:choice>
</xsd:complexType>
```

Gli elementi <logicalOps>3.2.1, <comparisonOps>3.2.1 e <spatialOps>3.2.1 sono riferimenti a gruppi per gli operatori logici, spaziali e di confronto. Gli operatori logici possono essere utilizzati per combinare gli operatori spaziali e di confronto nell'espressione di filtro. L'elemento <_id> ,invece ,è il riferimento per gli identificatori di un oggetto 3.2.1.

Spatial operators

Un operatore spaziale determina se i suoi argomenti geometrici rispettano la relazione spaziale dichiarata. L'operatore restituisce TRUE se la relazione spaziale è soddisfatta ,in caso contrario, restituisce FALSE.

Gli operatori spaziali codificati sono: Disjoint ,Touches ,Within ,Overlaps ,Crossesm ,Intersects ,Contains ,DWithin ,Beyond ,BBOX.
Si appresta a definire il solo elemento <BBOX> dato l'uso nella fase implementativa del progetto, è definito come un modo conveniente e più compatto di codifica del rettangolo di selezione basato sulla geometria gml:Envelope, è equivalente al funzionamento spaziale <Not> <Disjoint> ... </Disjoint> </Not> il che significa che l'operatore <BBOX> deve individuare tutte le geometrie che spazialmente interagiscono nella finestra. Se l'elemento opzionale <propertyName> non è specificato, il servizio deve determinare quale struttura è la chiave spaziale e applicare l'operatore BBOX di conseguenza.

Comparison operators

Un operatore di confronto viene utilizzato per formare espressioni che restituiscono il confronto matematico tra due argomenti. Se gli argomenti soddisfano il confronto, l'espressione restituisce TRUE. In caso contrario, restituisce FALSE.
Gli operatori di confronto sono: PropertyIsEqualTo ,PropertyIsNotEqualTo ,PropertyIsLessThan ,PropertyIsGreaterThan ,PropertyIsLessThanOrEqualTo ,PropertyIsGreaterThanOrEqualTo ,PropertyIsLike ,PropertyIsNull ,PropertyIsBetween.

Logical operators

Un operatore logico può essere usato per combinare una o più espressioni condizionali. L'operatore logico AND restituisce TRUE se tutte le espressioni combinate restituiscono TRUE. l'operatore OR restituisce TRUE se una delle espressioni combinate restituiscono TRUE. L'operatore NOT inverte il valore logico di un'espressione.
Gli elementi <And> ,<Or> e <Not> possono essere usati per combinare espressioni logiche in modo da formarne altre più complesse.

Object identifiers

L'identificatore di un oggetto serve a rappresentare un identificatore univoco per un'istanza di un oggetto all'interno del contesto del servizio web. Questa specifica non definisce un elemento specifico per identificare gli oggetti, ma definisce invece l'elemento astratto <_id> come un elemento per poi essere sostituito ed utilizzato alla definizione di un nuovo elemento identificatore per i tipi di oggetto specifico.
L'elemento <FeatureId> introdotte nella versione 1.0.0 identifica le istanze a partire dalla versione dei documenti GML2 utilizzando l'attributo "FID". Con

la versione GML 3.0, l'attributo FID è stato deprecato in favore di uno nuovo "gml:id" che può essere utilizzato per identificare gli elementi di tipo complesso GML in aggiunta alle funzioni, quali geometrie, topologie e gli attributi complessi. Un nuovo elemento viene quindi introdotto <GmlObjectId> per identificare gli elementi in versione GML3. L'elemento <FeatureId> viene mantenuto per retrocompatibilità.

Expressions

Un'espressione è una combinazione di uno o più simboli che restituiscono il valore booleano unico TRUE o FALSE.
Un'espressione può essere formata con gli elementi:

Arithmetic operators <Add>, <Sub>, <Mul>, <Div> :

Gli elementi sono utilizzati per codificare le operazioni aritmetiche fondamentali di addizione, sottrazione, moltiplicazione e divisione. Gli operatori aritmetici sono operatori binari e accettano due argomenti e restituiscono un unico risultato.

<PropertyName> :

L'elemento è usato per codificare il nome di qualsiasi proprietà di un oggetto. Il nome della proprietà può essere usato in espressioni scalari o spaziali per rappresentare il valore di quella proprietà per una particolare istanza di un oggetto.

<Literal> :

Si definisce con questa modalità la codifica del filtro con valori letterali. Un valore letterale è una parte di una comunicazione o di espressione che deve essere utilizzato esattamente come è specificato, piuttosto che come una variabile o un altro elemento.
L'elemento <literal> è usato per codificare i valori scalari e geometrici.
I valori letterali geometrici devono essere codificati come contenuto dell'elemento <literal>, secondo gli schemi GML [2].

<Function> :

Si definisce la codifica di funzioni a valore singolo utilizzando l'elemento <Function>. Una funzione è una procedura chiamata che esegue un calcolo distinto. Una funzione può accettare zero o più argomenti come input e genera un unico risultato.
Una funzione è composta dal nome della funzione, codificata con il nome dell'attributo, e zero o più argomenti contenuti all'interno dell'elemento <Function>.

L'elemento <expression> è un elemento astratto, il che significa che in realtà non esiste e il suo unico scopo è quello di agire come segnaposto per gli elementi e le combinazioni di elementi che possono essere utilizzati per formare espressioni.

3.3 OpenGIS® Styled Layer Descriptor

La possibilità di ricevere mappe o informazioni sui contenuti della mappa non è solo l'unica caratteristica degli standard definiti dall'OGC e visti nelle sezioni precedenti. Finora si è analizzato solo il modo di inviare richieste e ricevere risposte da parte di un server georeferenziale, ma non si è descritto come si deve operare per visualizzare l'informazione con un determinato tematismo scelto dall'utente. Non essendo stato ideato per questo caso, gli standard WMS e WFS non comprendono le direttive necessarie per creare tematismi, per questo motivo OGC ha implementato un altro standard denominato Styled Layer Descriptor (SLD).
Come si intuisce dal nome, con questo linguaggio è possibile creare tematizzazioni di punti, linee, poligoni, etichette di testo e molto altro ancora. SLD è quindi una sofisticata specifica OGC per la vestizione di layer vettoriali e raster. La tematizzazione avviene grazie all'utilizzo di elementi di tipo XML definiti tramite un XML Schema, ognuno con i propri parametri e le proprie funzionalità, tra cui la possibilità di utilizzare alcuni parametri dei CSS. L'SLD è un linguaggio molto flessibile, anche se nella sua generalità risulta di difficile comprensione infatti non ha una sintassi semplificata per i tipi di rendering più comuni. Una volta presa padronanza dello strumento si possono realizzare stili piuttosto sofisticati che riescono a soddisfare sia le specifiche GML che le specifiche Filter.
Attualmente esistono due versioni di SLD che vengono utilizzate per creare le tematizzazioni, ovverosia la 1.0.0 e la 1.1.0. Di norma la versione usata dalla maggior parte degli OpenGIS Web Server è la 1.0.0, a differenza della 1.1.0 ancora poco supportata.
Innanzitutto bisogna sottolineare che lo standard SLD è intrecciato di default allo standard WMS. Infatti, se si utilizza la direttiva GetMap di WMS si nota come lo styling SLD possa essere utilizzato in tre diversi modi:

- Il Client interagisce con il Server WMS attraverso il protocollo HTTP GET. Il file SLD viene utilizzato in modalità remota grazie all'aggiunta del parametro SLD=url_file_sld& nella query;

- Il Client usa sempre HTTP GET ma inserisce il contenuto XML dell'SLD direttamente nella query attraverso il parametro SLD_BODY (con dovute codifiche);
- Il Client interagisce con il server WMS attraverso HTTP POST, con la richiesta GetMap codificata in XML includendo anche il corpo dell'SLD;

Il terzo metodo è tecnicamente superiore rispetto agli altri due, purtroppo però non è implementato da quasi tutte le tecnologie lato server disponibili. L'utilizzo del secondo metodo, che è un compromesso tra il primo e il terzo metodo può comportare dei problemi nell'eccessiva lunghezza dell'URL. Di norma si utilizza il primo metodo, che è il più efficiente. Come annunciato precedentemente, lo standard Styled Layer Descriptor non è altro che una definizione di parametri grafici basati sul linguaggio XML. Gli elementi utilizzabili sono stati definiti attraverso l'utilizzo dell'SLD Schema (ovverosia un semplice XML Schema) che ne definisce la grammatica. Tali elementi XML, che verranno analizzati, sono:

- StyledLayerDescriptor (elemento radice)3.3.1
- Named Layer 3.3.1
- User-Defined Layer 3.3.1
- User-Defined Style 3.3.1
- FeatureTypeStyle 3.3.1
- Rule 3.3.1
- Symbolizer: poligoni 3.3.1

3.3.1 Elementi SLD

Un documento SLD è definito come una sequenza di styled layers. La radice StyledLayerDescriptor è definita dalla seguente frammento di XML Schema:

```
<xs:element name="StyledLayerDescriptor">
  <xs:complexType>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="sld:NamedLayer"/>
      <xs:element ref="sld:UserLayer"/>
    </xs:choice>
    <xs:attribute name="version" type="xs:string"
                  use="required"/>
```

```
  </xs:complexType>
</xs:element>
```

L'attributo *version* fornisce la versione del documento SLD, per facilitare la compatibilità con i documenti statici archiviati in diverse versioni della specifica. La stringa nel formato "x.y.z", è la stessa che nelle altre specifiche dei web server definite dall'OpenGIS. L'attributo è obbligatorio.
Gli styled layers possono corrispondere a *NamedLayer* o layer definiti dall'utente (*UserLayer*), che sono descritti nelle sezioni successive. Ci può essere un numero qualsiasi di entrambi i tipi di styled layer, anche nessuno, in qualsiasi ordine. L'ordine a cui i layer compaiono nel documento SLD sarà l'ordine che gli styled layers sono presentati, con i successivi styled layers saranno renderizzati rispetto al styled layers precedente.

Named Layers

Un "layer" è definito come un insieme di caratteristiche che possono essere potenzialmente di vari tipi di elementi. Un named layer è un livello che si può accedere da un web server OpenGIS utilizzando un nome noto.

User-Defined Layers

Oltre a utilizzare i named layers, è anche utile per essere in grado di definire layer personalizzati per il rendering. Dal momento che un layer è definito come un insieme di caratteristiche potenzialmente di tipo misto, l'elemento UserLayer deve fornire i mezzi per identificare le caratteristiche per essere utilizzato. Tutte le feature che devono essere rese per essere poi prelevate da una Web Feature Server (WFS) o un Web Coverage Service.

User-Defined Styles

Uno stile definito dall'utente consente lo styling della mappa per essere definito esternamente dal sistema e per essere passati in giro in un frammento interoperabile in formato XML. The per l'elemento SLD UserStyle lo schema è il seguente:

```
<xs:element name="UserStyle">
  <xs:complexType>
  <xs:sequence>
    <xs:element ref="sld:Name" minOccurs="0"/>
```

```
    <xs:element ref="sld:Title" minOccurs="0"/>
    <xs:element ref="sld:Abstract" minOccurs="0"/>
    <xs:element ref="sld:IsDefault" minOccurs="0"/>
    <xs:element ref="sld:FeatureTypeStyle"
                maxOccurs="unbounded"/>
  </xs:sequence>
  </xs:complexType>
</xs:element>

<xs:element name="Title" type="xs:string"/>
<xs:element name="Abstract" type="xs:string"/>
<xs:element name="IsDefault" type="xs:string"/>
```

Un UserStyle è allo stesso livello semantico di un NamedStyle utilizzati nel contesto di un WMS. Gli elementi Name, Title e Abstract consentono ad uno stile definito dall'utente di essere identificati e sono opzionali. Il nome dato è equivalente al nome di uno stile denominato WMS e viene usato come riferirimento allo stile ed identifica lo stile chiamato. Il titolo è una breve descrizione leggibile per lo stile che possono essere visualizzati in un elenco, e l'Abstract è una descrizione più precisa che può essere di pochi paragrafi.
Un UserStyle può contenere uno o più FeatureTypeStyles che permettono il rendering delle caratteristiche di tipi specifici.

FeatureTypeStyles

Il FeatureTypeStyle definisce lo stile che deve essere applicato ad una singola feature type di un layer. Esso è definito come segue:

```
<xs:element name="FeatureTypeStyle">
  <xs:complexType>
  <xs:sequence>
    <xs:element ref="sld:Name" minOccurs="0"/>

    <xs:element ref="sld:Title" minOccurs="0"/>
    <xs:element ref="sld:Abstract" minOccurs="0"/>
    <xs:element ref="sld:FeatureTypeName" minOccurs=
       "0"/>
    <xs:element ref="sld:SemanticTypeIdentifier"
                minOccurs="0" maxOccurs="unbounded"/
                   >
    <xs:element ref="sld:Rule" maxOccurs="unbounded"
       />
```

```
  </xs:sequence>
  </xs:complexType>
</xs:element>
```

L'elemento FeatureTypeStyle identifica la separazione esplicita nella SLD tra il trattamento di 'layer' e il trattamento delle feature type specifica. Il concetto di 'layer' è unica per WMS e SLD, ma le feature sono usate più in generale, come in WFS e GML, in modo esplicito questa separazione è importante.
Come in UserStyle, le FeatureTypeStyle possono avere un Name, Title, e Abstract.
Il FeatureTypeName identifica il tipo specifico di feature ed è lo stile ad essa associata. Il FeatureTypeStyle contiene uno o più elementi Rule che consentono il rendering condizionale.

Rules

Le Rules sono utilizzate per raggruppare le istruzioni per il rendering da condizioni sulle proprietà delle feature e dalla scala della mappa. Un esempio del formato di una Rule è mostrato nel seguente frammento XML Schema:

```
<xs:element name="Rule">
  <xs:complexType>
  <xs:sequence>
    <xs:element ref="sld:Name" minOccurs="0"/>
    <xs:element ref="sld:Title" minOccurs="0"/>
    <xs:element ref="sld:Abstract" minOccurs="0"/>
    <xs:element ref="sld:LegendGraphic" minOccurs="0
       "/>
    <xs:choice minOccurs="0">
      <xs:element ref="ogc:Filter"/>
      <xs:element ref="sld:ElseFilter"/>
    </xs:choice>
    <xs:element ref="sld:MinScaleDenominator"
                minOccurs="0"/>
    <xs:element ref="sld:MaxScaleDenominator"
                minOccurs="0"/>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="sld:LineSymbolizer"/>
      <xs:element ref="sld:PolygonSymbolizer"/>
      <xs:element ref="sld:PointSymbolizer"/>
      <xs:element ref="sld:TextSymbolizer"/>
      <xs:element ref="sld:RasterSymbolizer"/>
```

```
    </xs:choice>
  </xs:sequence>
  </xs:complexType>
</xs:element>
```

Solo una singola feature può essere all'interno di una Rule.
Gli elementi Filter e ElseFilter di una Rule permettono la selezione delle feature come descritto in 3.2.
L'elemento filtrante ha un significato relativamente semplice. La sintassi dell'elemento filtrante è definito nella nello standard [3]. I filtri possono essere utilizzati in diversi modi ed applicabili allo stesso FeatureTypeStyle possono sovrapporsi in termini di funzionalità e selezionate di ogni Rule.
Se una Rule non ha alcun elemento di filtrante, la condizione della Rule è sempre valida per tutte le feature sono accettate e vengono inserite nello stile dalla Rule.

Symbolizers

All'interno di una Rule, per le condizioni per gli stili agli elementi vengono utilizzati i Symbolizers. Un symbolizer descrive il modo di apparire una feature su una mappa. Il symbolizer non descrive solo la forma che dovrebbe apparire, ma anche le propriet grafiche come il colore e l'opacit. Attualmente, sono definiti cinque tipi di symbolizers:

- Line
- Polygon
- Point
- Text
- Raster

In questa sezione verr solo descritto il tipo Poligon usato per il progetto.

Polygon Symbolizer Un PolygonSymbolizer viene utilizzato disegnare un poligono (o altre geometrie di tipo area), compresi il suo riempimento al suo interno e ed il suo bordo(contorno). Essa ha la seguente definizione:

```
<xs:element name="PolygonSymbolizer">
  <xs:complexType>
  <xs:sequence>
    <xs:element ref="sld:Geometry" minOccurs="0"/>
```

```
      <xs:element ref="sld:Fill" minOccurs="0"/>
      <xs:element ref="sld:Stroke" minOccurs="0"/>
    </xs:sequence>
    </xs:complexType>
</xs:element>
```

L'elemento Geometry definisce la geometria da utilizzare per lo styling. L'unico metodo disponibile per la definizione di una geometria fare riferimento a una propriet di geometria utilizzando l'elemento ogc:PropertyName(definiti dal WFS). Il riempimento(Fill) e il tratto(Stroke) sono contenuti nel PolygonSymbol.

3.4 OpenGIS® CityGML

CityGML nasce come metodo generale per la strutturazione di modelli per ambienti urbani con scopi di visualizzazione, comprendendo oltre alle geometrie degli oggetti anche le loro proprietà semantiche e tematiche.

Definisce classi e relazioni per gli oggetti topografici più rilevanti nei modelli di città e regioni riguardanti le loro proprietà geometriche, topologiche, semantiche e il loro aspetto esterno.

Sono incluse generalizzazioni gerarchiche tra classi tematiche, aggregazioni, relazioni tra oggetti e proprietà spaziali. Le informazioni tematiche messe a disposizione fanno si che CityGML non sia semplicemente un formato per lo scambio e la visualizzazione di dati 3D dei modelli urbani, ma permette di usare tali modelli per analisi più sofisticate in ambiti applicativi diversi: simulazioni, data mining urbano, informazioni tematiche.

CityGML è un modello di dati aperto e basato su XML per memorizzare e scambiare dati spaziali 3D. Si tratta di uno schema basato sulla versione 3.1.1 di Geography Markup Language (GML3), standard internazionale per lo scambio di dati territoriali rilasciato dall'Open Geospatial Consortium (OGC) e la ISO TC211.

L'obiettivo di CityGML è quello di raggiungere una definizione comune delle entità di base, attributi e le relazioni di un modello 3D di una città. Questo è particolarmente importante per quanto riguarda la manutenzione e sostenibilità dei modelli 3D, consentendo il riutilizzo degli stessi dati in diversi campi applicativi.

Le parti principali di questo linguaggio sono il modello geometrico e il modello tematico. Il primo rappresenta tutte le informazioni di tipo geometrico e topologico in tre dimensioni degli oggetti del modello, mentre il secondo include informazioni di tipo semantico, che aiutano l'utente a stabilire le relazioni

tra i vari oggetti del modello, e utilizza il modello geometrico per modellare diversi oggetti tematici, ad esempio: "*Digital Terrrain model*", vegetazione [*vegetation*] (oggetti isolati e a allo stesso tempo una loro aggregazione), bacini idrici [*water bodies*], rete stradale[*transportation facilities*] e arredo urbano[*city furniture*].

Tutto ciò che non è esplicitamente modellato dallo standard è modellabile attraverso il concetto di oggetti generici e da attributi. Oggetti che hanno stessa forma e che appaiono molte volte anche in posizioni diverse, come gli alberi, possono essere modellati come prototipi e usati pù volte nel modello.Inoltre il concetto di *TerrainIntersectionCurve* è introdotto per integrare oggetti 3D con il *Digital Terrain Model* per avere le posizioni corrette al fine di evitare ad esempio edifici galleggianti e affossamenti nel terreno.

CityGML distingue cinque diversi livelli di dettaglio (LOD), in cui gli oggetti, all'aumentare del livello di dettaglio, diventano sempre più dettagliati sia dal punto di vista geometrico che tematico. Inoltre, gli oggetti possono avere delle referenze esterne corrispondenti ad oggetti in dataset esterni. Infine, agli oggetti CityGML è possibile assegnare un aspetto esteriore (appearances). Le apparenze non sono solo proprietà visive ma rappresentano anche proprietà osservabili arbitrarie delle superfici quali rumore, l'inquinamento o problemi strutturali indotti, ad esempio, da un terremoto.

Per una trattazione accurata di veda [1].

3.4.1 Caratteristiche generali

Modularisation

Il modello CityGML consiste nella definizioni di classi per i più importanti tipi di oggetti del modello cittadino 3D. Queste classi sono state individuate per essere necessarie ed importanti in molte applicazioni per diverse aree.

Il modello CityGML è scomposto in *core module* e *thematic extension modules*. Il *core module* comprende i concetti e le componenti di base di CityGML e, quindi, deve essere implementato da qualsiasi sistema. Basati sul *core module*, ogni estensione copre una tematica specifica di un campo del modello virtuale 3D della città. CityGML introduce le seguenti undici moduli tematici estensione: *Appearance*, *Building*, *CityFurniture*, *CityObjectGroup*, *Generics*, *LandUse*, *Relief*, *Transportation*, *Vegetation*, *WaterBody*.

Le implementazioni di CityGML possono supportare qualsiasi combinazione dei moduli di estensione in combinazione con il *core module*. Tali combinazioni di moduli sono denominati *CityGML profiles*. Pertanto, CityGML consente di implementare anche solo parzialmente il modello globale CityGML in modo valido.

Multi-scale modelling (5 levels of detail, LOD)

CityGML supporta cinque diversi livelli di dettaglio (LOD). All'aumentare del livello di dettaglio gli oggetti diventano sempre più particolareggiati sia per quel che riguarda la geometria sia per la tematica.

I diversi LOD sono:

- LOD0 : modello regionale (2.5D modello di terreno);
- LOD1: città/modello del sito (modello di blocco con o senza tetti);
- LOD2: città/modello del sito (texture di tetti e delle facciate);
- LOD3: città/modello del sito (modello architetturale dettagliato);
- LOD4: modello dell'interno (navigazione all'intero dell'edificio).

Ogni file CityGML può, ma non deve necessariamente, contenere rappresentazioni multiple per ogni oggetto in un diverso livello di dettaglio simultaneamente.

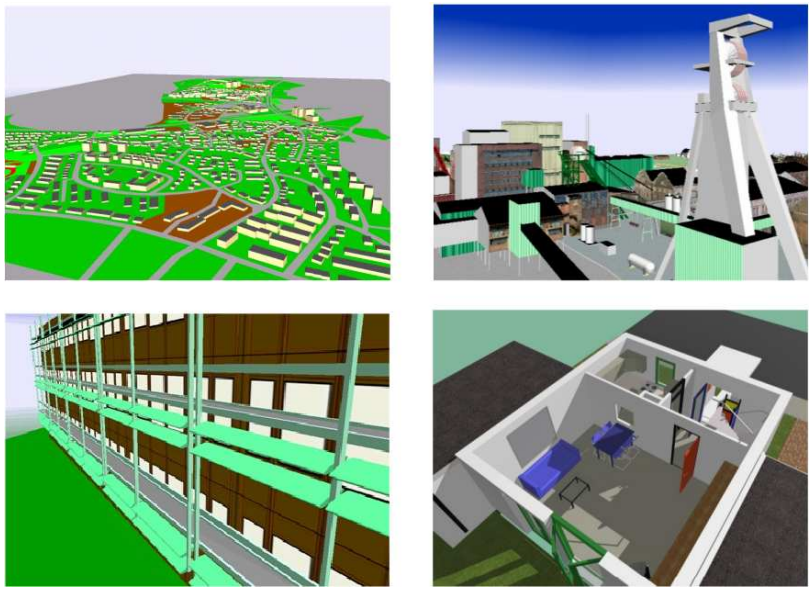

Figura 3.1: Esempio di LOD

City object groups

Il concetto di raggruppamento in CityGML permettere l'aggregazione di oggetti secondo criteri definiti dall'utente, e per rappresentare e trasferire queste aggregazioni come parte di un modello unico. Un gruppo può contenere altri gruppi come membri, permettendo raggruppamento nidificato di profondità arbitraria. Il concetto raggruppamento è espresso dall'estensione *CityObjectGroup* di CityGML.

Prototypic objects / scene graph concepts

In CityGML oggetti con forma uguale come gli alberi, semafori, segnali stradali, ecc. possono essere rappresentati come **prototipi** che vengono istanziati più volte in luoghi diversi. La geometria del prototipi è definita in un sistema di coordinate locali. Ogni istanza è rappresentata da un riferimento al prototipo, un punto base del sistema di coordinate di riferimento e una matrice di trasformazione che facilita il ridimensionamento, rotazione, e la traduzione del prototipo. Il principio è adottato dal concetto di scena utilizzati negli standards della computer graphics come VRML e X3D. Poiché GML3 non fornisce il supporto per la scena, questo è implementato come estensione di GML3.

3.4.2 modellazione geometrica e semantica

Uno dei più importanti principi di progettazione di CityGML è la coerenza tra la modellazione semantica e le proprietà geometriche/topologiche.

A livello semantico gli oggetti del mondo reale sono rappresentati da 'cose' come: edifici, muri, finestre o stanze. La loro descrizione include: attributi, relazioni ed aggregazioni gerarchiche tra oggetti. Quindi la parte di relazioni tra gli oggetti può essere derivata solo a livello semantico e non geometrico. A livello spaziale, gli oggetti geometrici sono assegnati ad oggetti che rappresentano la loro locazione ed estensione nello spazio.

Quindi il modello è compostato da due livelli gerarchici:

- semantico
- geometrico

in cui gli oggetti corrispondenti sono legati tra loro da relazioni di tipo gerarchico.

Il vantaggio di questo approccio consiste nel fatto che è possibile effettuare una navigazione in entrambe le gerarchie e tra le gerarchie arbitrariamente, per poter eseguire query tematiche e/o geometriche.

Capitolo 4

Tecnologie utilizzate

4.1 PostgreSQL/PostGIS

PostgreSQL™ è un sistema di gestione di database relazionale ad oggetti (ORDBMS) basato su POSTGRES, Version 4.2™, sviluppato alla "University of California", nel dipartimento di informatica Berkeley. POSTGRES proponeva molti concetti che diventarono disponibili solo in alcuni sistemi di database commerciali molto più tardi.
PostgreSQL™ è il discendente open-source di quel codice originale Berkeley. Supporta una parte molto grande dello standard SQL e offre molte altre funzionalità:

- query complesse
- chiavi esterne
- trigger
- viste
- integrità transazionale
- controllo concorrente multiversione

Inoltre, PostgreSQL™ può essere esteso dall'utente in molti modi, per esempio aggiungendo nuovi tipi di dato, funzioni, operatori, funzioni aggregate, metodi di indice, linguaggi procedurali.
Data la licenza libera, PostgreSQL™ può essere usato, modificato e distribuito da chiunque gratuitamente per qualsiasi scopo, sia esso privato, commerciale, o accademico.

PostGIS è l'estensione spaziale del server PostgreSQL che introduce i tipi di dato geometrico e le funzioni per lavorare con essi. Inoltre fornisce le definizioni dei sistemi di coordinate (derivati dalla classificazione EPSG) per eseguire trasformazioni tra geometrie materializzate in sistemi di riferimento diversi.
Attenendosi ai formati descritti da Open Geospatial Consortium , per garantire trasparenza ed interoperabilità, vengono usati i tipi: POINT, MULTIPOINT, LINESTRING, MULTILINESTRING, POLYGON, MULTIPOLYGON, GEOMETRYCOLLECTIONS (più le estensioni SQL-MM CIRCULARSTRING, COMPOUNDCURVE, CURVEPOLYGON, MULTICURVE, MULTISURFACE) con estensione XYZ,XYM, XYZM.
Per ogni strato informativo occorre definire una colonna che contenga le coordinate, ed un sistema che gestisca in maniera differenziata le tabelle che contengono dati geometrici di tipo differente. Occorre inoltre introdurre un insieme di vincoli per verificare che i dati inseriti siano congruenti: ad esempio dovrebbe essere sempre verificata la bi o tri-dimensionalità nello stesso dataset. Interrogazioni veloci alle tabelle sono la ragione d'essere dei database (assieme al supporto per le transazioni). I database si differenziano proprio per la maggior robustezza e prestanza degli indici. Le elaborazioni GIS avvengono usando la sintassi SQL sui costrutti "spaziali". Esistono comunque funzioni che costruiscono immediatamente tutta la struttura necessaria alla gestione dei dati territoriali. Analogamente le interrogazioni al database avvengono utilizzando "SQL query" utilizzando combinazioni di dati e funzioni e di test vero/falso. Per comprendere come funzioni una query spaziale occorre tenere presenti due cose:

1. esistono gli indici spaziali;

2. le interrogazioni spaziali sono molto dispendiose in termini di calcolo se effettuate su un gran numero di entità geometriche.

Gli indici spaziali esistono per migliorare l'efficienza delle query. I dati geografici vengono "aggregati" e "amministrati" in gruppi spaziali distinti. In questo senso gli indici sono lossy: sono definiti per semplificare e nella semplificazione perdono informazioni. Ad esempio: spesso si vuole utilizzare l'operatore intersezione && che testa se il rettangolo che circoscrive le entità geometriche ne intersechi altre. Questa funzione è ottimizzata per l'utilizzo degli indici spaziali: serve a scremare i dati per esegure una ricerca più raffinata usando le funzioni Distance, Intersects, Contains, Within... Le funzioni che utilizzano l'indice spaziale servono per identificare le geometrie che possono soddisfare ad una condizione. Le funzioni spaziali testano la condizione esattamente.
PostgreSQL utilizza 3 tipi di indici: B-Tree, R-Tree e GiST (Generalized Search

Tree). Per PostgreSQL, R-Tree ha l'inconveniente di NON essere "null safe": non possono essere immagazzinate geometrie NULLE, degeneri o tipi diversi all'interno della stessa colonna. Sinteticamente, PostGIS manipola dati "congruenti".

4.2 Geoserver

Geoserver è un software open source, Java-based, che permette ai suoi utenti di visualizzare, modificare e condividere dati geospaziali attraverso l'utilizzo di protocolli definiti dall' OGC.
GeoServer è nato nel 2001 da "The Open Planning Project" (TOPP), un progetto con sede a New York ideato per creare una suite di strumenti che rendessero l'attività del governo più trasparente. Con GeoServer nacque anche il progetto GeoTools, una libreria Java sviluppata per consentire a GeoServer di lavorare con vari formati vettoriali e geoDBMS.
GeoTools toolkit è tuttora sfruttato da GeoServer ed è caratterizzato da un'architettura modulare che permette di aggiungere funzionalità in modo agevole, è rilasciato sotto GNU Lesser General Public Licence (LGPL) e contiene metodi standard compliant per la manipolazione di dati spaziali. Contestualmente allo sviluppo di GeoServer infatti, OGC stava lavorando allo standard Web Feature Service (WFS), un protocollo per rendere i dati spaziali direttamente disponibili sul web usando GML (Geographic Markup Language, la grammatica XML definita sempre dall'OGC per specificare le caratteristiche di oggetti geografici).
Altri progetti che nacquero proprio per rafforzare le funzionalità di GeoServer sono:

- Refractions Research, che realizzò PostGIS, un database spaziale che abilita GeoServer alla connessione ai database.

- MetaCarta, ideò invece Open Layers, una libreria Java Script open source. Tale libreria è tuttora integrata in GeoServer e rilasciata sotto la licenza BSD-style; è in continuo sviluppo e rende la generazione delle mappe e la visualizzazione delle stesse nei moderni web browser, facile e veloce.

La prima versione di GeoServer (1.0) è stata rilasciata nell'ottobre del 2001. SourceForge.net (al momento il più grande sito di sviluppo e controllo di software open source), in quel mese, registrò circa 500 download del pacchetto GeoServer.
Poco dopo il rilascio della prima versione, che implementava solo la specifica

WFS definita da OGC, un programmatore di un'azienda commerciale spagnola sviluppò il supporto per la specifica Web Map Service (WMS), un protocollo per la creazione e visualizzazione di mappe create a partire da dati spaziali.
Successivamente altri progetti hanno contribuito ad incrementare le funzionalità di questo server geospaziale. Nel 2005 la compagnia italiana GeoSolutions sviluppò il supporto per lo standard Web Coverage Services (WCS) e per la grafica raster che rappresenta le immagini descrivendole come una griglia di pixel opportunamente colorati (contrapposta alla grafica vettoriale che invece descrive le immagini attraverso primitive geometriche che definiscono punti, linee e poligoni ai quali possono essere attribuiti colori e sfumature).
Nell'agosto del 2007 è stata poi rilasciata la versione 1.5.3 e sono stati registrati approssimativamente 8500 download. Al crescere di questi ultimi e dei progetti sviluppati intorno a GeoServer, corrispondeva la crescita del numero di utenti e sviluppatori.
Il successo di GeoServer è dovuto proprio al fatto che, essendo un progetto open source, la sua gestione è sempre stata delegata a tutta la comunità (formata da soggetti provenienti da diverse organizzazioni) e mai a una singola organizzazione.
La versione corrente è la 2.0.2, rilasciata il 24 Maggio 2010 ed è già disponibile la versione 2.1 Beta (4 Settembre 2010).
Lo sviluppo di GeoServer continua per rendere i dati spaziali accessibili a tutti. In questo senso, sta lavorando con Google per rendere i propri dati accessibili e ricercabili attraverso Google Map. Presto, ricercare dati spaziali diventerà quindi facile come cercare una pagina web su Google.

4.2.1 Struttura catalogo

La directory dati di GeoServer è la posizionata nel file system dove GeoServer memorizza tutta la sua configurazione.
Questa configurazione definisce come: quali dati GeoServer fornisce, dove sono collocati i dati, come i servizi (WMS e WFS) interagiscono con i server e con i dati. La directory contiene anche una serie di file di supporto utilizzati da GeoServer per vari scopi. In generale gli utenti non ha bisogno di conoscere la struttura delle directory dei dati, ma è una buona idea definire una directory esterna per i dati, per rendere più semplice l'aggiornamento, la produzione e la ricerca.

La struttura della directory a questo punto è quasi esclusivamente essere di interesse per gli sviluppatori. Nelle versioni precedenti gli utenti spesso modificavano direttamente i dati e le varie configurazioni di Geoserver nella directory, con la nuova versione 2.x.x questo è possibile con le REST API, ed è l'unica

opzione consigliata.
La struttura di una directory dati di GeoServer:

file .xml I file xml a questo livello servono per salvare le informazioni sui servizi e le varie opzioni globali.

File	Descrizione
global.xml	Contiene le impostazioni tra i servizi, comprese le informazioni sui contatti, le impostazioni JAI, set di caratteri.
logging.xml	Specifica la registrazione, il luogo, e gli accessi al std out.
wcs.xml	Contiene i metadati e le varie impostazioni per il servizio WCS.
wfs.xml	Contiene i metadati e le varie impostazioni per il servizio WFS
wms.xml	Contiene i metadati e le varie impostazioni per il servizio WMS

Tabella 4.1: File di configurazione di primo livello

workspaces La directory workspaces contiene i metadati relativi ai "layers" che sono pubblicati da GeoServer. Ogni layer avrà un file *layer.xml* ad esso associati ed un xml o un file "*featuretype.xml*" a seconda se si tratta di un raster o vettoriale.

data Da non confondere con "GeoServer data directory", in questa directory possono essere memorizzati effettivamente i dati, è comunemente utilizzata per archiviare shapefiles e file raster, ma può essere utilizzata per tutti i dati basati su file.
Il principale vantaggio di immagazzinare file dati all'interno è la portabilità, in modo da essere utilizzato da tutta l'utenza come path unico senza ulteriori modifiche.

demo La directory demo contiene i file che definiscono le richieste per agli esempi contenuti nel Sample Request Tool (voce Demos del menù principale di Geoserver)

geosearch La directory GeoSearch contiene le informazioni per le "regionation" dei file KML.

gwc Questa directory contiene la cache creata dal servizio integrato GeoWebCache.

layergroups Contiene informazioni sulle configurazioni dei layer-groups.

palettes La directory viene usata per memorizzare i vari insiemi di colori per le immagini. Le "*Image palettes*" sono utilizzate da GeoServer per WMS come modo per ridurre le dimensioni delle immagini prodotte, pur mantenendo la qualità dell'immagine stessa.

security La directory contiene tutti i file utilizzati per configurare la protezione di GeoServer, per esempio i ruoli di accesso ai dati.

styles La directory contiene una serie di file per gli stili di Styled Layer Descriptor(SLD). Per ogni file in questa directory vi è una corrispondente voce in *catalog.xml*:
<style id=point_style file=default_point.sld/>

templates La directory contiene i file utilizzati da GeoServer per i template. I template sono utilizzati per personalizzare i risultati delle varie operazioni di GeoServer.

user_projections La directory contiene un unico file chiamato *epsg.properties* che viene utilizzato per definire i sistemi di riferimento spaziali spaziali che non fanno parte del database ufficiale EPSG.

4.2.2 Servizi e dati supportati

Dati

Questa sezione descrive quale tipologia di dati che GeoServer è in grado di leggere e di accedere. GeoServer permette il caricamento e e la fruizione dai servizi dei seguenti formati di dati di default:

- Vector data formats
  - Shapefiles
  - PostGIS databases
  - External WFS layers
  - Java Properties files
- Raster data formats
  - ArcGrid
  - GeoTIFF
  - Gtopo30

  - ImageMosaic
  - WorldImage

Altre tipologie di dati richiedono l'uso di estensioni da parte di GeoServer, alcune di esse sono disponibili sulla pagina di download di GeoServer.

Servizi

GeoServer pubblica dati utilizzando protocolli standard stabiliti dal Open Geospatial Consortium. Il Web Feature Service (WFS) permette di effettuare richieste di *geographical feature data*(dati vettoriali). Il Web Map Service (WMS) consente di effettuare richieste di immagini generate da dati geografici. Infine, il Web Coverage Service (WCS) permette di effettuare richieste di dati di copertura (dati raster). Questi servizi sono il cuore di GeoServer.

- **WFS**
  GeoServer fornisce il supporto per Open Geospatial Consortium (OGC) Web Feature Service (WFS), versione 1.0 e 1.1. Questo è uno standard per ottenere dati vettoriali attraverso il web. L'utilizzo di un WFS rende possibile ii client l'interrogazione della struttura dei dati e dei dati effettivi.
  GeoServer è l'implementazione di riferimento per entrambe le versioni 1.0 e 1.1 dello standard, completamente implementate ogni parte del protocollo. Questo comprende le operazioni di base di GetCapabilities, DescribeFeatureType e GetFeature, come pure le operazioni più avanzati delle transazioni, LockFeature e GetGmlObject. Il WFS di GeoServer, inoltre, è integrato con il sistema di sicurezza GeoServer, in modo da limitare l'accesso ai dati e alle transazioni.

- **WMS**
  GeoServer fornisce il supporto per Open Geospatial Consortium (OGC) Web Map Service (WMS) versione 1.1.1, questo è uno standard per la generazione di mappe sul web.
  GeoServer supporta completamente ogni parte dello standard, ed è certificata conforme dalla suite di test dell'OGC. Esso comprende una grande varietà di opzioni di rendering e di etichettatura, ed è uno dei più veloci server WMS sia per i dati raster che vettoriali.
  Si appoggia pienamente anche lo standard (SLD) Styled Layer Descriptor, ed utilizza infatti i file SLD delle relativi per il rendering.

- **WCS**
  GeoServer fornisce il supporto per Open Geospatial Consortium (OGC) Web Coverage Service (WCS), versioni 1.0 e 1.1, il WCS può essere pensato come l'equivalente di Web Feature Service, ma per i dati raster, invece di dati vettoriali. Esso consente di ottenere le informazioni di copertura, non solo l'immagine. GeoServer è l'implementazione di riferimento per il WCS 1.1.

Capitolo 5

Servizio WFS per CityGML

5.1 Stato dell'arte

Nel contesto attuale le soluzioni esistenti sono proprietarie (Snowflake CityGML-WFS) o sono ferme nel loro sviluppo (deegree-3D), ad uno stadio iniziale e parziale. Inoltre, come già detto in ambito Open Geospatial Consortium (OGC) si è ancora nella fase di discussion paper ma solo come standard per la visualizzazione e non come estrazione di singole feature.

Finora non è ancora iniziata la discussione per uno standard WFS 3D.

Di seguito vengono presentate le soluzioni esistenti ed i probabili standard futuri:

Snowflake CityGML WFS Snowflake CityGML WFS è stato creato sulla base di Snowflake Go Publisher. GO Publisher è un traduttore di dati da database relazionali in XML. Per funzionare quindi, i dati sono stati caricati dapprima in un database Oracle utilizzando uno strumento chiamato GO Loader. Questo strumento ha una capacità simile alla traduzione di GO Publisher, ma traduce da GML ai modelli relazionali.
I dataset CityGML per lo scenario di prova di questo servizio sono stati creati dall'istituto Forschungszentrum Karlsruhe, convertendo i modelli codificati in Industry Foundation Classes (IFC) in modelli CityGML.
Una volta messo in atto l'application server, Snowflake CityGML WFS permette di elaborare le richieste per le operazioni WFS: GetCapabilities, DescribeFeatureType e GetFeature. Al ricevimento della richiesta GetFeature, GO Publisher traduce la richiesta in una query SQL utilizzando una traduzione configurata in fase di sviluppo. La query SQL risultante contiene tutte le condizioni della richiesta WFS effettuata, viene eseguita sul database Oracle il cui risultato verrà poi tradotto da GO Publisher in GML. Il GML risultante viene poi

trasmesso al client. I dati possono essere restituiti in un flusso compresso, se richiesto.

Forschungszentrum Karlsruhe Web Feature Services L'Istituto di Informatica Applicata (IAI), del Forschungszentrum Karlsruhe (Karlsruhe Research Center) ha istituito un server WFS aperto. Il server WFS utilizza come motore deegree con la sua componente 3D:
Deegree 3D è una soluzione software open source per le elaborazione delle informazioni geospaziali 3D. Si basa su standard OGC e comprende la gestione, l'accesso e la rappresentazione dei dati 3D.
Possono essere importati dati sotto forma di modello CityGML e VRML ma l'unica opzione di esportazione è in CityGML. Per gestire i dati utilizza il database PostGIS, per l'accesso ad essi le interfacce WFS, WMS, WCS, WPVS in versione draft per fornire le varie componenti e per visualizzare tramite un'interfaccia leggera le varie viste di immagini 3D. Lo sviluppo è fermo, le funzioni testate, sono ad uno stato iniziale e parziale

W3DS Il Web 3D Service (W3DS) è un servizio di rappresentazione dei geodati tridimensionali quali modelli per il paesaggio, urbani, edifici con texture, vegetazione.
I dati sono forniti come scene composte da elementi di visualizzazione, ottimizzati per il rendering in tempo reale a frame rate elevati. Le scene 3D possono essere mostrate interattivamente utilizzando direttamente browser con il plugin 3D, o possono essere caricati in applicazioni per la visualizzazione virtuale. Il W3DS è in grado di gestire insiemi di dati con una vasta gamma di scale, e da più livelli di dettaglio per ciascun oggetto, quindi aumentare notevolmente le prestazioni senza sacrificare la qualità, questo lo pone molto simile ad un Web Map Service (WMS). I formati utilizzati per la codifica delle scene 3D sono progettati per le reti a banda limitata, come Internet, ed avere un rendering efficace in tempo reale in modo da evitare un sovraccarico prodotto ad esempio da XML. A causa del formato utilizzato i contenuti espressi dal W3DS non sono limitati ad oggetti statici, ma possono anche includere animazioni e altri effetti visivi, nonché i comportamenti predefiniti da interazioni con l'utente.

WTS/WPVS/WVS Un WPVS è un servizio web in grado di generare viste prospettiche del terreno, ad esempio, immagini di un certo terreno/area (può contenere oggetti tridimensionali, ad esempio edificio o alberi) da un punto di vista richiesto. Per questa finalità il WPVS ha bisogno di elaborare e visualizzare diversi tipi di dati geospaziali, estratti da diversi set di dati pre-configurati. Questi dataset possono essere servizi web OGC remoti o installati localmente.

Il Web View Service (WVS) è un servizio rappresentazione di geo-dati tridimensionali quali modelli di paesaggio, di città, di vegetazione, o di infrastrutture di trasporto. Il WVS incapsula l'intero processo di rappresentazione a lato server e fornisce a più strati di immagini renderizzate delle scene 3D all'utente. Il WVS estende il Web Terrain Service (WTS) ed il Web Perspective View Service (WPVS), in un certo senso, il WVS può essere considerato come la controparte 3D per il ben noto Web Map Service (WMS).
Il WVS è progettato per superare la visualizzazione ristretta e capacità di interazione degli approcci basati sul WTS/WPVS. Come una estensione a questi, il WVS fornisce ulteriori dati geometrici e tematici, come ad esempio informazioni di profondità e sull'identità degli oggetto.
Inoltre, il WVS permetta operazioni per il recupero di informazioni su oggetti visualizzati in specifici posizioni delle immagini, come le misure ed il supporto di navigazione avanzata.

5.2 Progettazione e Sviluppo

5.2.1 Architettura

L'applicazione sviluppata è una applicazione "web-based". Gli utenti interagiscono con il sistema utilizzando un browser o un client.
L'architettura Web 3D GIS (Figura 5.1) è basata su servizi web OGC, e si configura come un'estensione del server geospaziale.
Per la trasmissione dei dati geografici tra i livelli dell'architettura applicativa si usa uno stream formattato in CityGML, che può ereditare la interoperabile ed estensibilità di GML.
Di seguito vengono descritti brevemente i vari layer dell'architettura presa in considerazione.

Client

La sua funzione è di trasmettere le richieste e ricevere i dati, formattati in CityGML, dal client al server e viceversa utilizzando il protocollo WFS, tramite metodi HTTP GET/POST.

Web Server

La sua funzione è principalmente di fare da un contenitore al web server GIS (Geoserver), e trasmettere le risposte alle richieste da parte del client. Nel nostro caso è stato scelto Tomcat.
Apache Tomcat (o semplicemente Tomcat) è un web container open source

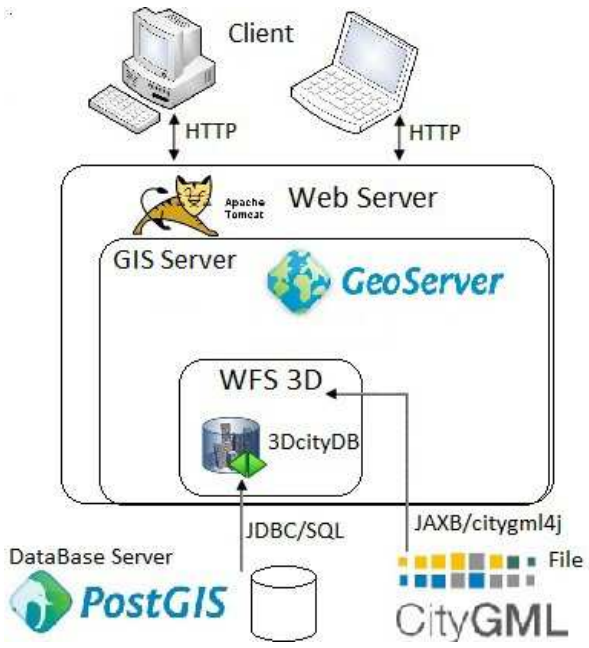

Figura 5.1: Architettura del servizio implementato

sviluppato dalla Apache Software Foundation. Implementa le specifiche JavaServer Pages (JSP) e Servlet di Sun Microsystems, fornendo quindi una piattaforma per l'esecuzione di applicazioni Web sviluppate nel linguaggio Java. La sua distribuzione standard include anche le funzionalità di web server tradizionale, che corrispondono al prodotto Apache.

GIS Server

La sua funzionalità è di creare le connessioni al database, l'analisi e lo scambio di dati geospaziali tramite varie operazioni e servizi standard. Diversi prodotti open source (Geoserver, Deegree, MapServer ...) effettuano queste funzioni. Si è scelto Geoserver, 4.2 poiché è l'implementazione di riferimento e fornisce supporto per le specifiche Open Geospatial Consortium (OGC) Standard.

WFS3D

Al fine di dare supporto al geo-dati 3D, è necessario ampliare le funzioni del web-server GIS con l'aggiunta del modello CityGML, creare quindi delle specifiche funzionalità per questi dati, al fine di fornire funzionalità 3D al web service.

Questo sarà l'oggetto del lavoro ed in Figura 5.2 è raffigurato il diagramma del servizio WFS per rendere disponibili le informazioni geografiche nella rete.

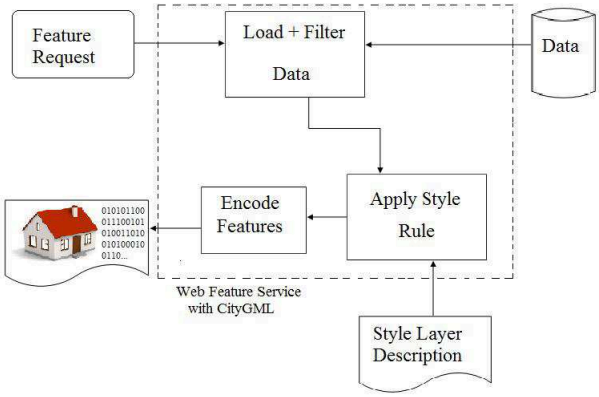

Figura 5.2: Workflow del servizio

3DcityDB

3DcityDB è lo strumento software per l'importazione e l'esportazione di un dataset CityGML per il geo-database 3D con uno schema relazionale appropriato, lo strumento è considerato parte integrante di una soluzione globale per una efficiente elaborazione, conservazione e recupero dei modelli della città 3D. Il tool di import/export usa una architettura software multithread per elaborazioni dei dati ad alte prestazioni, si compone di tre parti: il processo di binding con la definizione dello schema XML di CityGML in un oggetto JAVA, il processo di importazione ed esportazione dei dati. In questo lavoro si utilizzeràà solo la parte di binding ed esportazione.
Il workflow è realizzato come una catena di processi che unisce diversi pool di thread, ognuno dei quali ricopre un diverso step del processo: interrogazione del database in base ai criteri(filtri) definiti dall'utente, ricevute le informazioni esse vengono mappate nei corrispondenti oggetti JAXB i quali sono trasformati(marshall) in eventi SAX per essere scritti nelle istanze del documento CityGML. Per maggiori informazioni [1].

Database

Le sue funzioni sono principalmente di archiviazione ed accesso ai dati spaziali e non spaziali.
Per poter memorizzare i dati all'interno della base di dati è necessario creare uno schema relazionale del database derivato dal diagramma UML delle classi

di CityGML. Da un'analisi condotta dal IGGSS per lo sviluppo del proprio tool è emerso che per l'elaborazione e la memorizzazione dei modelli nel database, è sufficiente uno schema semplificato, per migliorare le performance in tempo di elaborazione dei dati. Con questa struttura il DataBase PostgreSQL/PostGIS viene poi utilizzato tramite l'applicativo 3DCityDB per importare ed esportare feature da esso. Per maggiori informazioni [1].

5.2.2 Interfaccia web

L'interfaccia web di amministratore è il tool per la configurazione di tutti gli aspetti di GeoServer, dall'aggiunta dei dati alla configurazione dei servizi. Nella maggior parte delle installazioni, GeoServer partirà come un server web su localhost sulla porta 8080 accessibile dal seguente URL:
http://localhost:8080/geoserver/web
Anche se questo URL può variare a seconda di come e dove GeoServer è stato installato.

Welcome a Geoserver

Accedendo con un qualsiasi browser web all'indirizzo sopra citato, se correttamente configurato, verrà visualizzata una pagina di benvenuto nel browser.
La pagina di benvenuto contiene i collegamenti alle varie aree della configurazione di GeoServer. La pagina fornisce anche il login a GeoServer, Figura 5.3.

Figura 5.3: Login per la parte amministrativa

Questa misura di sicurezza impedisce a persone non autorizzate di apportare modifiche alla configurazione di GeoServer. Il nome utente e la password di default sono *admin* e *geoserver*, modificabili tramite il file in *security/users.properties* nella directory dati.
Avvenuto il login la pagina si presenterà con maggiori opzioni rispetto alla precedente: Figura 5.4. Dalla pagina di amministrazione di entra all'interno della configurazione del servizio WFS 3D, il quale si suddivide in 3 parti: Figura 5.5.

- Service Metadata
- DataStore

- Feature

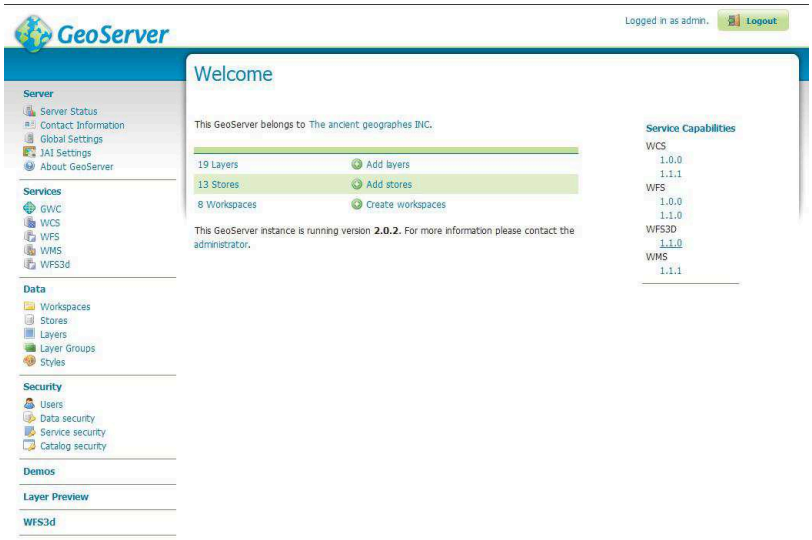

Figura 5.4: Pagina di Welcome dopo essersi loggati

Figura 5.5: Homepage per il servizio WFS 3D

WFS3D - Service Metadata

In questa sezione vengono brevemente descritti i parametri del servizio implementato.Figura 5.6.

Figura 5.6: Pagina di configurazione del servizio.

Enabled
: Specifica se il servizio WFS3D è abilitato o disabilitato. Quando è disattivato, le richieste al servizio non verranno elaborate;

Manteiner
: Nome dell'organizzazione per il mantenimento del servizio;

Online Resource
: Definisce l'URL HTTP del servizio. In genere l'URL della "home page"

(Obbligatorio);

Title
Titolo per identificare il servizio agli utenti. (Obbligatorio);

Abstract
Fornisce una descrizione con ulteriori informazioni sul servizio;

Keywords
Elenco di parole chiavi connesse con il servizio in aiuto nella ricerca;

Service Level
Specifica il livello del servizio WFS3D. La Service Level è una maschera che indica quali operazioni del WFS3D sono disponibili.
Basic: Offre la ricerca ed il recupero dei dati con la funzione GetCapabilities, DescribeFeatureType e le operazioni GetFeature;

CityGML
Il servizio di default risponde con un documento CityGML e un predefinito stile di Spatial Reference System (SRS), in formato URN. Questi formati possono essere:
Normal: Restituisce numero tipico EPSG (EPSG:XXXX);
XML: Restituisce un URL che identifica il codice EPSG (http://www.opengis.net/gml/srs/epsg.xml#XXXX);
URN: Restituisce la formattazione SRS delimitata da colonne (urn:x-ogc:def:crs:EPSG:XXXX).

WFS3D - DataStore

Il DataStore si connette alla sorgente dati contenente le informazioni di oggetti 3D seguendo il modello CityGML. La sorgente dati può essere un file o un database PostGIS strutturato come 3DGeoDabase per CityGML.Figura 5.7

Figura 5.7: Menù per la selezione della tipologia dello store.

Editing a Store:

Per poter visualizzare e modificare uno store, selezionare il nome dello store dal menù a tendina, dopo il quale si presenterà una pagina con le informazioni dello store scelto, dipende dal formato specifico scelto e salvare la nuova configurazione.Figura 5.8.

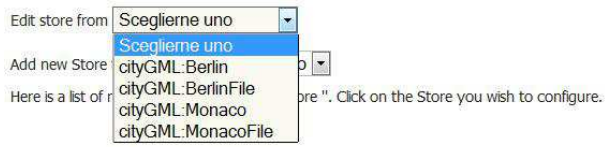

Figura 5.8: Menù per la selezione alla modifica degli store esistenti.

Adding a Store:

Per aggiungere uno nuovo store al servizio, usare il menù a tendina per selezionare il tipo di store da aggiungere: *PostGIS*, *CityGMLfile*, dopo il quale si presenterà una pagina con le informazioni dello store scelto, dipende dal formato specifico scelto, da inserire. Figura 5.9.

I campi da inserire/modificare nello store sono:

Basic Store Info

Name:
è il nome dello store, elencato nella pagina precedente;

Workspace:
viene impostato di default CityGML e non è modificabile

Description
è facoltativa e viene visualizzato solo nell'interfaccia di amministrazione;

Connection Parameters

URL:[solo per il tipo CityGMLfile] posizione del file CityGML;

dbtype
tipo di database,impostazione predefinita con PostGIS;

host
nome dell'host dove risiedede il database;

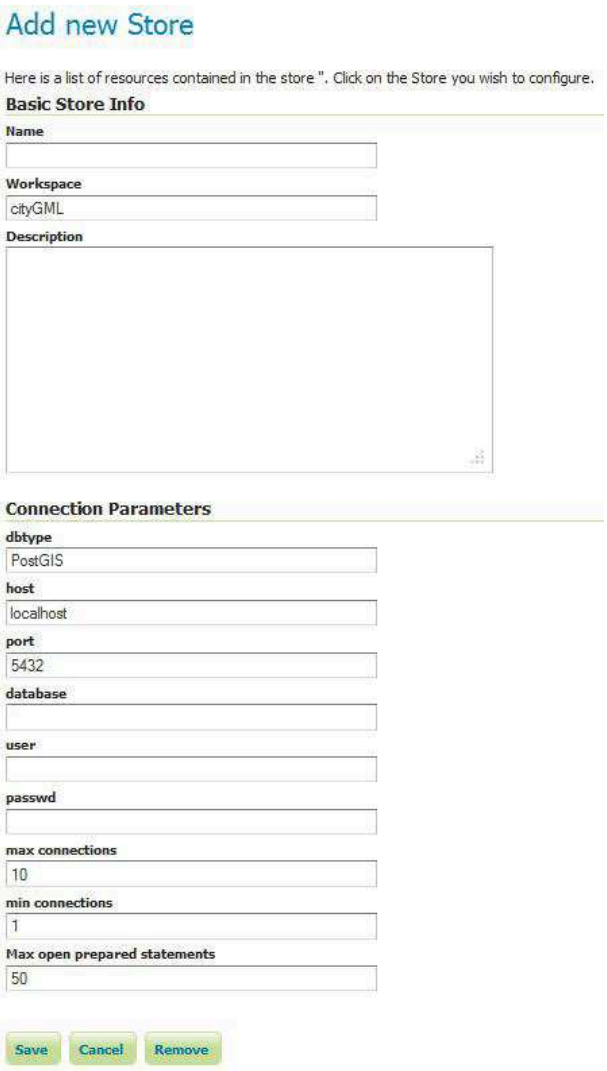

Figura 5.9: configurazione parametri store.

port
: numero di porta per la connessione all'host;

database
: nome del database noto nell'host;

user
: nome utente per la connessione al database;

passwd
: Password associata all'utente;

max connections
numero massimo di connessioni aperte al database;

min connections
numero minimo di connessioni;

max open prepared statements
numero massimo di prepared statement.

Terminato l'inserimento e/o modifica dei parametri di configurazione dello store il sistema passa direttamente alla pagina di scelta di inserimento delle feature al servizio associate allo store.

WFS3D - Feature

In questa sezione si descriverà il metodo di inserimento delle feature associato ad uno store inserito precedentemente, la scelta di questo viene effettuata tramite un menù a tendina, figura AddFeature.
Una volta scelto lo store si scelgono quali Feature si vogliono rendere disponibili tramite il servizio: All, Building, WaterBody, LandUse, Vegetation, Transportation, ReliefFeature, CityFurniture, GenericCityObject, CityObjectGroup, per il tipo CityGMLfile è resa disponibili la sola feature All. Figura 5.10
Ogni Feature da inserire ha dei parametri associati:Figura 5.11

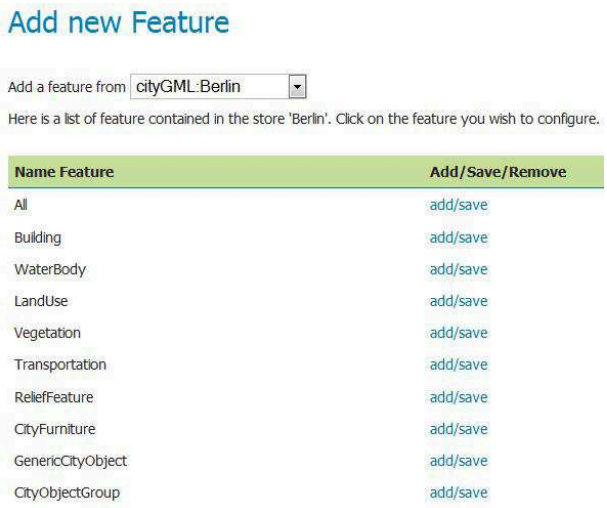

Figura 5.10: Scelta Feature dallo store.

Name
identificativo utilizzato per riferire al servizio la Feature, preimpostato come: ”*nomeDataStore-nomeFeature*”;

Title
descrizione leggibile per l’utente;

Abstract
descrizione narrativa con ulteriori informazioni;

Coordinate Reference System - srs
definisce come i dati georeferenziati sono localizzati sulla superficie terrestre, il servizio deve sapere qual’è il sistema di riferimento spaziale(SRS) associato al dato, questo può venir calcolato direttamente tramite l’apposita funzione ”*Computate from native Data*”;

Bounding Boxes -BBOX
Il bounding box determina l’estensione dei dati sotto forma di rettangolo, è possibile calcolarli tramite l’apposita funzione ”*Computate from native bounds*”

5.2.3 Implementazione

Per implementare il nuovo modulo in Geoserver in modo che sia possibile rendere disponibile ed elaborare informazioni spaziali 3D in formato CityGML si è utilizzato il framework del servizio OWS di Geoserver in modo che sia integrato con tale applicativo.

Questa sezione spiega come è stato creato il servizio per GeoServer. Il servizio fornisce tre operazioni denominate “GetCapabilities”, “DescribeFeatureType”, “GetFeature”. Il risultato delle operazioni segue lo standard WFS.

Setup

La prima operazione da eseguire per creare un plugin per Geoserver è quella di creare un progetto maven:

- creare una nuova cartella per il plugin denominata “*WFS3D*”;
- creare il maven pom all’interno denominato “*pom.xml*”;
- creare la struttura per i sorgenti del progetto “*src/main/java*”;

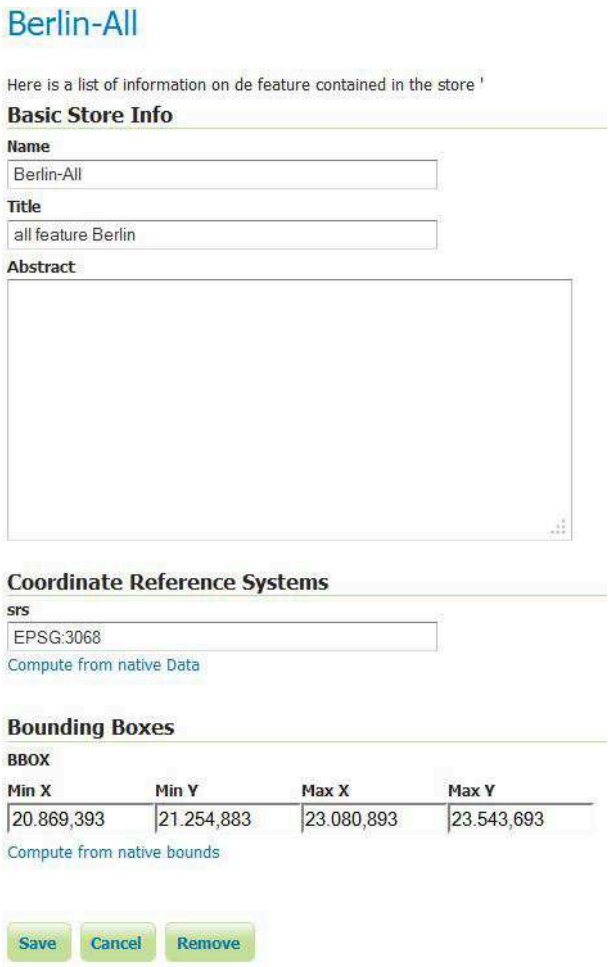

Figura 5.11: Configurazione parametri Feature.

- creazione della classe per il servizio, al suo interno saranno implementate le operazioni da effettuare "*DefaultWebServiceFeature3D*" (L'elenco di parametri viene automaticamente fornito da org.geoserver.ows.Dispatcher);

- Creare un "*applicationContext.xml*" dichiarando le classi come bean per spring.

Per installare il modulo creato si esegue la sua compilazione attraverso maven:

mvn install

Se questa è andata a buon fine viene creato il jar in "target/wfs3d-1.0.jar" e copiato nella cartella "WEB-INF/lib" nella directory di installazione di GeoServer, successivamente deve essere riavviato per il corretto caricamento del modulo appena creato.

Interfaccia Web

In questa sezione viene spiegato come è stata aggiunta l'interfaccia web al servizio tramite i Wicket-UI.

- si aggiunge al file *pom.xml* le dipendenze associate per il web;
- vengono create tante classi quante le pagine web da voler utilizzare, queste saranno estensioni della classe base per tutte le pagine di Geoserver *"org.geoserver.web.GeoServerBasePage"*;
- creazione delle pagine html di presentazione con i vari wicket creati nella classe associata;
- creazione del file per l'internalizzazione (i18n) "*GeoServerApplication.properties*";
- aggiunta in "*applicationContext.xml*" dei bean per linkare le varie pagine alle classi associate ed alle proprietà descritte nelle punti precedenti;

Classi Principali

package org.geoserver.wfs3d: in questo package viene implementato il core del servizio con la sua classe "*DefaultWebServiceFeature3D*" la quale tramite il dispacther di Geoserver vengono elaborate le richieste prodotte dal client per poi dare in output la risposta. Per ogni operazione del servizio, GetCapabilities, DescribeFeatureType, GetFeature, sono stati creati gli omonimi metodi i quali richiamano a loro volta la classe omonima per effettuare le varie elaborazione per la risposta alla chiamata.
All'interno di questo package vengono anche implementate le classi per l'aggiunta del servizio WFS3D a Geoserver in modo che interagisca nel miglior modo possibile con questo.

- *GetCapabilities*
  classe utilizzata per rispondere all'omonimo servizio GetCapabilities, scrive in output un file xml al cui interno vengono inserite le varie informazioni del servizio, queste vengono richiamate tramite il catalogo di Geoserver nel quale sono salvate tramite le apposite pagine web di amministrazione le parti riguardanti l'identificazione, del provider, le operazioni possibili, la lista delle Feature disponibili per CityGML ed i filtri ad esso implementati;
- *DescribeFeatureType*
  la suddetta classe scrive in outputStream lo schema XSD di CityGML, per il quale il suddetto servizio è stato implementato;

- *GetFeature*
  si può ritenere la classe più importante delle tre implementate poichè tramite questa il servizio rende disponibile la porzione di CityGML richiesta. Questa classe fa dapprima un parsing della richiesta del client in modo da capire quali filtri e stili applicare alle Feature da esportare, se la Feature richiesta è situata in un Database PostGIS questo viene impostato per l'esportazione tramite l'applicativo/libreria 3DCityDB ed interfacciato tramite la classe "Gconfig" situata nel package "org.geoserver.wfs3d.data", al contrario se la Feature è un file CityGML questo viene immediatamente proposto alla post-elaborazione per la tematizzazione. Al termine della post-elaborazione il file risposta prodotta viene inviata al client con il formato richiesto: *cityGML-gzip* o *cityGML*, output compresso o meno.

package org.geoserver.wfs3d.data: Il package in questione viene usato dal servizio per l'implementazione delle operazioni di pre e post elaborazione dei documenti CityGML. Le classi in esso contenute sono richiamate dalla classe GetFeature, per effettuare il parser del documento SLD per l'eventuale tematizzazione e filtraggi della richiesta, se la Feature è archiviata tramite database viene effettuato l'interfacciamento con questi tramite l'apposita libreria 3DCityDB, ed infine la post-elaborazione nella quale si apportano le modifiche per rendere l'informazione leggibile da una fonte esterna.

- *sld*
  Classe utilizzata per effettuare il parser del documento SLD. Questa classe ha il compito di tenere memorizzate le varie tematizzazioni contenute in un file scritto secondo lo standard Style Layer Description tramite un oggetto di tipo theme.
  Le regole di colorazione devono essere inseriti come "PolygonSymbolizer" con il nome della geometria(es. "RoofSurface") e il colore in codice esadecimale (es. #FF0000 per il colore rosso), infine sono stati implementati i valori per i filtri sugli id ed il filtro "IsGreaterThan" per colorare gli edifici più alti di un certo valore passato.

- *Elaboration*
  Classe di post-elaborazione del servizio. Aggiunge le appaerence all'output andando a selezionare le varie superfici secondo le impostazioni salvate precedentemente nella richiesta. A questo poi viene aggiunto il colore assegnato ed inserito nel CityGML.

- *Gconfig*
  È la classe di interfacciamento per l'esportazione delle informazioni dal

database PostGIS. Con questa classe si impostano i valori di connessione i vari filtri (BoundingBox, Filter ID, GML name e la selezione delle singole Feature). Queste impostazioni verrano poi utilizzate da 3DCityDB per effettuare la connessione con il database ed per effettuare le query a seconda dei filtri richiesti.

- *org.geoserver.wfs3d.data.filter.FilterQuery*
  questa classe effettua il parser dei filtri della richiesta del client per settarli tramite la classe "Gconfig", in questo modo vengo impostati direttamente alla lettura di essi.

package org.geoserver.wfs3d.web: Il package viene utilizzato per l'implementazione delle pagine web del servizio. Per la creazione delle pagine web si il framework Apache Wicket . Wicket è uno dei framework di programmazione per sviluppare applicazioni Web disponibili per Java, tra i suoi punti di forza e innovazioni rispetto agli altri web framework vi sono: un eccellente supporto built-in ad AJAX, la capacità di modificare la struttura delle pagine programmaticamente, perfetta separazione tra codice e templates (i templates non possono contenere codice), un approccio object-oriented, facile apprendimento (per developers Java-oriented), inoltre è l'unico web framework programmabile in puro codice Java. Le varie pagine sono descritte in 5.2.2.

package org.geoserver.wfs3d.xml: Questo package viene utilizzato seguendo il framework dell'OWS di Geoserver per l'implementazione del parser XML al servizio, vengono estese le interfacce di OWS per poi essere implementate per il parser delle richieste al servizio.
Queste classi servono per istanziare i vari bean di Spring in modo tale che vengano utilizzati dal dispatcher per analizzare un particolare valore di parametro di richiesta in HTTP. La ragione per analizzare il valore del parametro è che questi valori stanno vengono spediti/ricevuti sotto forma di String (in codifica UTF-8 o altro), ma prima di poter essere correttamente riconosciuti e utilizzati da GeoServer devono essere convertiti nel tipo appropriato di oggetti Java o tipi primitivi. Per esempio il parametro "*bbox*" deve essere convertito nell'oggetto envelope, e queste sottoclassi fanno esattamente questo lavoro.

Capitolo 6

Risultati ottenuti

Per effettuare il collaudo e le prove dell'implementazione del progetto è stato utilizzato la stessa macchina sia per il Database server, PostgreSQL/PostGIS, che per la parte relativa al server web, Tomcat - Geoserver, come client un computer utilizzando un browser web qualsiasi. Come esempi di prova per le varie richieste, il dataset possono essere sia file CityGML che li stessi già importati nel database tramite l'apposito tool, e successivamente questi dataset dovranno essere aggiunti al catalogo del servizio per essere disponibili via HTTP.

6.1 Esempi di richieste WFS 3D

Di seguito vengono riportati alcuni esempi di richieste WFS-3D, sia come una chiamata HTTP GET che come chiamata HTTP POST(tutte le richieste vengono effettuate in locale, 127.0.0.1):

GetCapabilities

Esempio operazione GetCapabilities del servizio.

HTTP GET :

http://localhost:8080/geoserver/wfs3d?
service=wfs3d&
version=1.1.0&
request=GetCapabilities

HTTP POST :

```
<?xml version='1.0' encoding='UTF-8'?>
<GetCapabilities
   service="WFS3D"
   xmlns="http://www.opengis.net/wfs"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-
      instance"\par
   xsi:schemaLocation="http://www.opengis.net/wfs␣
      http://schemas.opengis.net/wfs/1.1.0/wfs.xsd"/>
```

Risposta XML :

```
<wfs:WFS_Capabilities updateSequence="" version="
    1.1.0" xsi:schemaLocation="http://www.opengis.net
    /wfs␣http://localhost:8080/geoserver/schemas/wfs
    /1.1.0/wfs.xsd">
<ows:ServiceIdentification>
   ...
</ows:ServiceIdentification>
<ows:ServiceProvider>
   ...
</ows:ServiceProvider>
<ows:OperationsMetadata>
   ...
</ows:OperationsMetadata>
<wfs:FeatureTypeList>
   ...
</wfs:FeatureTypeList>
<ogc:Filter_Capabilities/>
   ...
</wfs:WFS_Capabilities>
```

DescribeFeatureType

Esempio operazione DescribeFeatureType del dataset di Berlino.

HTTP GET :

http://localhost:8080/geoserver/wfs3d?
service=wfs3d&
version=1.1.0&
request=DescribeFeatureType&
typeName=cityGML:Berlin-All

HTTP POST :

```
<?xml version='1.0' encoding='UTF-8'?>
<DescribeFeatureType
  version="1.1.0"
  service="WFS3D"
  xmlns="http://www.opengis.net/wfs"
  xmlns:topp="http://www.openplans.org/topp"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
     instance"
  xsi:schemaLocation="http://www.opengis.net/wfs␣
     http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
    <TypeName>cityGML:Berlin-All</TypeName>
</DescribeFeatureType>
```

Risposta schema XSD di cityGML :

```
!-- CityGML Version No. 1.0.0, May 19th, 2008-->
<!-- CityGML - GML3 application schema for the 3D
   city model of the Special Interest Group 3D (SIG
   3D) of GDI NRW-->
...
<xs:schema targetNamespace="http://www.opengis.net/
   citygml/1.0"
elementFormDefault="qualified"
attributeFormDefault="unqualified">
...
<!-- == -->
<!-- ==CityGML Core module == -->
<!-- == -->
<!-- ==Root Element: CityModel== -->
<!-- == -->
<xs:complexType name="CityModelType">
```

```
...
/xs:complexType>
<!-- == -->
<xs:element name="CityModel" type="CityModelType"
    substitutionGroup="gml:_FeatureCollection"/>
   ...
```

GetFeature

GetFeature con filtro Feature ID

Esempio operazione GetFeature del dataset di Berlino con filtro Multi Feature ID.

HTTP GET :
http://localhost:8080/geoserver/wfs3d?
service=wfs3d&
version=1.1.0&
request=GetFeature&
typeName=cityGML:Berlin-All&
FEATUREID=UUID_e34b7e76-8616-49ce-bb20-1cb7df2ac681, UUID_43e414e2-2ce0-4457-a0b5-9040607be75e, UUID_58bc1a18-61f3-48c1-905f-0e626f66f39d

HTTP POST :

```
<?xml version='1.0' encoding='UTF-8'?>
<wfs:GetFeature service="WFS3D" version="1.1.0"
   outputFormat="cityGML"
   xmlns:topp="http://www.openplans.org/topp"
   xmlns:wfs="http://www.opengis.net/wfs"
   xmlns:ogc="http://www.opengis.net/ogc"
   xmlns:gml="http://www.opengis.net/gml"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-
      instance"
   xsi:schemaLocation="http://www.opengis.net/wfs␣
      http://schemas.opengis.net/wfs/1.1.0/wfs.xsd">
   <wfs:Query typeName="cityGML:Berlin-All">
    <ogc:Filter>
     <ogc:FeatureId fid="UUID_e34b7e76-8616-49ce-bb20
        -1cb7df2ac681"/>
```

```
      <ogc:FeatureId fid="UUID_43e414e2-2ce0-4457-a0b5
         -9040607be75e"/>
      <ogc:FeatureId fid="UUID_58bc1a18-61f3-48c1-905f
         -0e626f66f39d"/>
    </ogc:Filter>
  </wfs:Query>
</wfs:GetFeature>
```

Risposta CityGML :

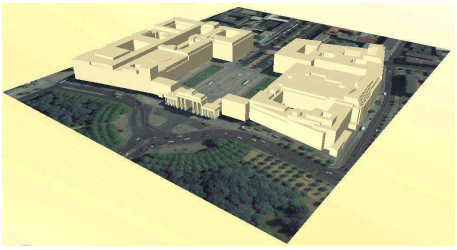

Figura 6.1: Dataset Berlin originale.

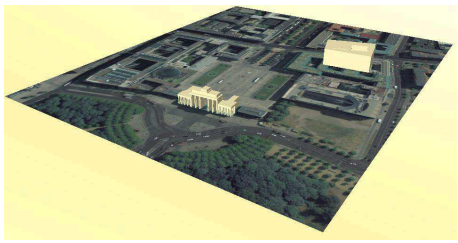

Figura 6.2: Dataset Berlin esportato con filtri.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes
   "?>
<CityModel ... >
    <gml:description>Export from WFS3D</
       gml:description>
    <cityObjectMember>
        <luse:LandUse gml:id="UUID_58bc1a18-61f3-48
           c1-905f-0e626f66f39d">
            <gml:name>Simple textured patch</
               gml:name>
            ...
            <app:appearance> ...
            <luse:lod2MultiSurface> ...
        </luse:LandUse>
    </cityObjectMember>
    <cityObjectMember>
        <bldg:Building gml:id="UUID_e34b7e76-8616-49
           ce-bb20-1cb7df2ac681">
            <gml:name>Britische Botschaft</gml:name>
```

```
            ...
        </bldg:Building>
    </cityObjectMember>
    <cityObjectMember>
        <bldg:Building gml:id="UUID_43e414e2-2ce0
            -4457-a0b5-9040607be75e">
            <gml:name>Brandenburger Tor</gml:name>
            ...
            <bldg:lod2Solid>...
            <bldg:address>...
        </bldg:Building>
    </cityObjectMember>
</CityModel>
```

GetFeature con SLD

Esempio operazione GetFeature con tematizzazione tramite file SLD (A.2) passato come URL,il quale colora di marrone i muri, di rosso i tetti e gli edifici superiori ad una data altezza di viola.

HTTP GET :
http://localhost:8080/geoserver/wfs3d?
service=wfs3d&
version=1.1.0&
request=GetFeature&
typeName=cityGML:MonacoFile-All&
sld=http://localhost:8080/geoserver/style.sld

Risposta CityGML :

Figura 6.3: Dataset Munich originale.

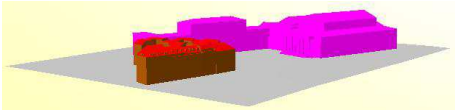

Figura 6.4: Dataset Munich con SLD file dato.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes
    "?>
```

```
<CityModel ... >
    <gml:description>Export from WFS3D</
       gml:description>
    <cityObjectMember>
        <bldg:Building>
        ...
    <app:appearanceMember>
        <app:Appearance>
        ...
   <app:appearanceMember>
        <app:Appearance>
            <app:theme>SLDstyle</app:theme>
            <app:surfaceDataMember>
                <app:X3DMaterial>
                    <app:diffuseColor>
                       0.5019607843137255
                       0.25098039215686274 0.0</
                       app:diffuseColor>
                    <app:target>#UUID_02c315a0
                       -2470-4b0d-af95-487e1cf7262b<
                       /app:target>
                    ...
                </app:X3DMaterial>
            </app:surfaceDataMember>
            <app:surfaceDataMember>
                <app:X3DMaterial>
                    <app:diffuseColor>1.0 0.0 1.0</
                       app:diffuseColor>
                    <app:target>#UUID_414cd86b-d859
                       -41d9-b18a-de873d8f27df</
                       app:target>
                    ...
                </app:X3DMaterial>
            </app:surfaceDataMember>
            <app:surfaceDataMember>
                <app:X3DMaterial>
                    <app:diffuseColor>1.0 0.0 0.0</
                       app:diffuseColor>
                    <app:target>#UUID_b4339341-97c0
                       -4e78-a023-11bceeb34d5d</
```

```
                        app:target>
                    ...
                </app:X3DMaterial>
            </app:surfaceDataMember>
            <app:surfaceDataMember>
                <app:X3DMaterial>
                    <app:diffuseColor>0.0 0.6 1.0</
                        app:diffuseColor>
                </app:X3DMaterial>
            </app:surfaceDataMember>
        </app:Appearance>
    </app:appearanceMember>
</CityModel>
```

Capitolo 7

Conclusioni

Gli obiettivi posti sono stati ampiamente raggiunti realizzando un servizio web come estensione di Geoserver, per gli oggetti in formato CityGML in analogia con il servizio Web Feature Service di OGC ed ampliandone le caratteristiche con funzioni di filtro e con la tematizzazione dinamica e personalizzata delle Feature tramite lo Style Layer Description.

Il servizio Web Feature Service è un'ottima soluzione per gli obiettivi del progetto. Le richieste vengono generate a livello di client e vengono inviate al server WFS. Quest'ultimo legge ed esegue la richiesta, ritornando i dati risultanti come dati vettoriali, ovvero in una serie di feature codificate nel formato CityGML. Un client abilitato alla decodifica di CityGML può quindi utilizzare i dati per l'esecuzione di query o analisi come se gli stessi fossero residenti sul suo client.

Sono state effettuati diversi test utilizzando vari dataset, reperibili da `http://www.citygml.org`, in diversi LOD (Level Of Detail) sia caricati nel database che direttamente su file CityGML verificando la correttezza del servizio creato.

7.1 Sviluppi futuri

Il plugin realizzato fornisce un'estensione del WFS per CityGML in modo "Basic", quindi effettua la sola lettura delle informazioni memorizzate, può essere ampliato nella modalità "Transaction", per consentire di effettuare oltre all'esportazione anche l'inserimento e la modifica delle informazioni archiviate.

Dato l'enorme onere computazionale e le grandi dimensioni delle informazioni geospaziali in formato CityGML, anche se in parte risolto comprimendo il flusso in output, si può ottimizzare l'estensione implementando una cache.

Inoltre, la tematizzazione allo stato attuale riguarda solo colori a tinta unita, sarebbe interessante sviluppare la tematizzazione delle feature di CityGML predisponendo, via documenti Style Layer Description (SLD), colori sfumati e soprattutto l'aggiunta di texture diverse da quelle originali, determinabili in relazione all'utilizzo del dato.

Appendice A

Schemi XML

A.1 Schema Web Feature Service

A.1.1 DescribeFeatureType schema

```
<xsd:element name="DescribeFeatureType" type="
   wfs:DescribeFeatureTypeType"/>
<xsd:complexType name="DescribeFeatureTypeType">
  <xsd:complexContent>
    <xsd:extension base="wfs:BaseRequestType">
      <xsd:sequence>
        <xsd:element name="TypeName" type="xsd:QName
           " minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="outputFormat" type="
         xsd:string" use="optional" default="text/
         xml;␣subtype=gml/3.1.1"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

A.1.2 GetFeaure schema request

```
<xsd:element name="GetFeature" type="wfs:
   GetFeatureType"/>
<xsd:complexType name="GetFeatureType">
  <xsd:complexContent>
    <xsd:extension base="wfs:BaseRequestType">
      <xsd:sequence>
```

```
        <xsd:element ref="wfs:Query" maxOccurs="
           unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="resultType" type="wfs:
         ResultTypeType" use="optional" default="
         results"/>
      <xsd:attribute name="outputFormat" type="xsd:
         string" use="optional" default="text/xml;
         subtype=3.1.1"/>
      <xsd:attribute name="maxFeatures" type="xsd:
         positiveInteger" use="optional"/>
      <xsd:attribute name="traverseXlinkDepth" type
         ="xsd:string" use="optional"/>
      <xsd:attribute name="traverseXlinkExpiry" type
         ="xsd:positiveInteger" use="optional"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:simpleType name="ResultTypeType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="results"/>
    <xsd:enumeration value="hits"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:element name="Query" type="wfs:QueryType"/>
<xsd:complexType name="QueryType">
  <xsd:sequence>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element ref="wfs:PropertyName"/>
      <xsd:element ref="ogc:Function"/>
    </xsd:choice>
    <xsd:element ref="ogc:Filter" minOccurs="0"
       maxOccurs="1"/>
    <xsd:element ref="ogc:SortBy" minOccurs="0"
       maxOccurs="1"/>
  </xsd:sequence>
  <xsd:attribute name="handle" type="xsd:string" use
     ="optional"/>
  <xsd:attribute name="typeName" type="wfs:
     TypeNameListType" use="required"/>
```

```
  <xsd:attribute name="featureVersion" type="xsd:
     string" use="optional"/>
  <xsd:attribute name="srsName" type="xsd:anyURI"
     use="optional"/>
</xsd:complexType>
<xsd:simpleType name="Base_TypeNameListType">
  <xsd:list itemType="QName"/>
</xsd:simpleType>
<xsd:simpleType name="TypeNameListType">
  <xsd:restriction base="wfs:Base_TypeNameListType">
    <xsd:pattern value="((\w:)?\w(=\w)?){1,}"/>
  </xsd:restriction>
</xsd:simpleType>
```

A.1.3 GetFeature schema response

```
<xsd:element name="FeatureCollection" type="wfs:
   FeatureCollectionType" substitutionGroup="gml:
   _FeatureCollection"/>
<xsd:complexType name="FeatureCollectionType">
  <xsd:complexContent>
    <xsd:extension base="gml:
       AbstractFeatureCollectionType">
      <xsd:attribute name="lockId" type="xsd:string"
         use="optional"/>
      <xsd:attribute name="timeStamp" type="xsd:
         dateTime" use="optional"/>
      <xsd:attribute name="numberOfFeatures" type="
         xsd:nonNegativeInteger" use="optional"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

A.1.4 GetCapabilities schema Response

```
<xsd:element name="WFS_Capabilites" type="wfs:
   WFS_CapabilitiesType" substitutionGroup="ows:
   Capabilites"/>
<xsd:complexType name="WFS_CapabilitiesType">
  <xsd:complexContent>
    <xsd:extension base="ows:CapabilitiesBaseType">
      <xsd:sequence>
```

```
        <xsd:element ref="wfs:FeatureTypeList"
           minOccurs="0"/>
        <xsd:element ref="wfs:
           ServesGMLObjectTypeList" minOccurs="0"/>
        <xsd:element ref="wfs:
           SupportsGMLObjectTypeList"/>
        <xsd:element ref="ows:Filter_Capabilities"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

A.2 Esempio style.sld

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<StyledLayerDescriptor version="1.0.0"
    xsi:schemaLocation="http://www.opengis.net/sld␣
       StyledLayerDescriptor.xsd"
    xmlns="http://www.opengis.net/sld"
    xmlns:ogc="http://www.opengis.net/ogc"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
       instance">
  <NamedLayer>
    <Name>Default</Name>
    <UserStyle>
      <Title>default style</Title>
      <Abstract>A sample style that just prints out
         a coulored feature</Abstract>
      <FeatureTypeStyle>
        <!-- Rule 1 -->
        <Rule>
          <Name>RoofSurface</Name>
          <Title>red roof</Title>
          <Abstract>A green line with a 2 pixel
             width</Abstract>
          <PolygonSymbolizer>
            <Geometry>
              <ogc:PropertyName>RoofSurface</
                 ogc:PropertyName>
            </Geometry>
```

```
      <Fill>
        <CssParameter name="fill">#FF0000</
           CssParameter>
      </Fill>
    </PolygonSymbolizer>
  </Rule>
  <!-- Rule 2 -->
  <Rule>
    <Name>WallSurface</Name>
    <Title>brown wall</Title>
    <Abstract>A green line with a 2 pixel
       width</Abstract>
    <PolygonSymbolizer>
      <Geometry>
        <ogc:PropertyName>WallSurface</
           ogc:PropertyName>
      </Geometry>
      <Fill>
        <CssParameter name="fill">#804000</
           CssParameter>
      </Fill>
    </PolygonSymbolizer>
  </Rule>
  <!-- Rule 3 -->
  <Rule>
    <Name>Window</Name>
    <Title>window blue</Title>
    <Abstract>A green line with a 2 pixel
       width</Abstract>
    <PolygonSymbolizer>
      <Geometry>
        <ogc:PropertyName>Window</
           ogc:PropertyName>
      </Geometry>
      <Fill>
        <CssParameter name="fill">#0099FF</
           CssParameter>
      </Fill>
    </PolygonSymbolizer>
  </Rule>
```

```
        <!-- Rule 4 -->
        <Rule>
          <Name>greatthan</Name>
          <Title>Filter PropertyIsGreaterThan</Title
             >
          <Abstract></Abstract>
          <Filter>
            <ogc:PropertyIsGreaterThan>
              <ogc:PropertyName>MeasuredHeight</
                 ogc:PropertyName>
              <ogc:Literal>30</ogc:Literal>
            </ogc:PropertyIsGreaterThan>
          </Filter>
          <PolygonSymbolizer>
            <Fill>
              <CssParameter name="fill">#FF00FF</
                 CssParameter>
            </Fill>
          </PolygonSymbolizer>
        </Rule>
        <!-- Rule 5 -->
        <Rule>
          <Name>FeatureId</Name>
          <Title>filter FeatureId</Title>
          <Abstract></Abstract>
          <Filter>
            <ogc:FeatureId fid="UUID_9a30a2dc-dffe
               -455a-a055-d5b2ace975b9"/>
          </Filter>
          <PolygonSymbolizer>
            <Fill>
              <CssParameter name="fill">#FF00FF</
                 CssParameter>
            </Fill>
          </PolygonSymbolizer>
        </Rule>
        </FeatureTypeStyle>
    </UserStyle>
  </NamedLayer>
</StyledLayerDescriptor>
```

Bibliografia

[1] P. Varagnolo, "Progettazione e sviluppo di un toolkit per la gestione di dati spaziali 3d nei formati standard ogc citygml e kml per il geodatabase opensource postgis.," Master's thesis, Università degli studi di Padova, aprile 2011.

[2] S. Cox, P. Daisey, R. Lake, C. Portele, and A. Whiteside, *OpenGIS® Geography Markup Language (GML)*. Open Geospatial Consortium Inc., 2004-02-07. Version: 3.1.1.

[3] P. A. Vretanos, *OpenGIS® Filter Encoding Implementation Specification)*. Open Geospatial Consortium Inc., 3 May 2005. Version: 3.1.1.

[4] L. Bianchini, "Progettazione di un tool open source per l'importazione, l'esportazione e la manipolazione di dati in formato standard citygml su database postgis," Master's thesis, Università degli studi di Padova, Ottobre 2009.

[5] G. Gröger, T. H. Kolbe, A. Czerwinski, and C. Nagel, *OpenGIS City Geography Markup Language (CityGML) Encoding Standard*. OGC, 2008-08-20. Version: 1.0.

[6] T. H. Kolbe, G. König, C. Nagel, and A. Stadler, *3DCityDB-Documentation*. Institute for Geodesy and Geoinformation Science, Technische Universität Berlin, 2009. v2.0.1.

[7] OpenGeo, "Geoserver." `http://geoserver.org`.

[8] SIG_3D and GDI_NRW, "Citygml." `http://www.citygml.org`.

[9] A. Stadler and T. H. Kolbe, "Spatio-semantic coherence in the integration of 3d city models," in *5th International Symposium on Spatial Data Quality - Modelling qualities in space and time*, 13-15 June 2007.

[10] A. Stadler, C. Nagel, G. König, and T. H. Kolbe, "Making interoperability persistent: A 3d geo database based on citygml," *Proceedings of the 3rd International Workshop on 3D Geo-Information*, p. 175, 2009.

[11] P. A. Vretanos, *Web Feature Service Implementation Specification*. Open Geospatial Consortium Inc., 3 May 2005. Version: 1.1.0.

[12] A. Whiteside, *OGC Web Services Common Specification*. Open Geospatial Consortium Inc., 2007-02-09. Version: 1.1.0.

[13] D. M. Lupp, *Styled Layer Descriptor profile of the Web Map Service Implementation Specification*. Open Geospatial Consortium Inc., 2007-06-29. version: 1.1.0(revision 4).